ABSTRACT  Session-based recommendations have attracted significant attention because of their broad application scenarios. Recently, graph neural networks (GNNs) have been employed in session-based recommendation because of their superior performance in representation learning compared with recurrent neural networks (RNNs). Although most existing GNN-based methods have made great achievements in this field, none of them emphasizes the importance of repeat recommendations, which has been an important component in session-based recommendation (e.g., people tend to browse product information repeatedly or revisit websites in a period of time). In this paper, we propose a novel model called ReGNN to combine a graph neural network with a repeat-exploration mechanism for better recommendations. Specifically, we dynamically process the item sequence of a session as a graph structure and capture the complex transitions between items by a GNN. Then, we formulate an exact session representation with the attention mechanism. Finally, the repeat-exploration mechanism is incorporated into the ReGNN to model the user’s repeat-exploration behavior patterns and make more accurate predictions. We conduct extensive experiments on two public datasets. The experimental results show that our proposed model ReGNN consistently outperforms other state-of-the-art methods.

INDEX TERMS Session-based recommender systems, graph neural networks, repeat-explore mechanisms, and attention mechanisms.

I. INTRODUCTION

Recommender systems play an important role in people’s daily lives. In many real-world applications, recommender systems help users alleviate the problem of information overload and recommend products that users are likely interested in. Most existing recommender systems are based on user historical interactions or other user information. However, in many cases, a user’s profile and past activities are not recorded constantly, which results in only a group of interactions that take place in a user session. To address the challenge above, session-based recommendation is proposed to predict the next item that a user may click or visit when just given the sequence of the user’s actions in the current session [1]–[3].

Some conventional methods have been proposed based on Markov chains (MC) [4], [5] to model users’ behaviors, which are only based on the previous item that a user visits. The primary problem of these methods is that if a user clicks an item by mistake or randomly, the performance of the model may be strongly influenced. In addition, it is difficult for MC-based methods to model high-order relationships. Later, Hidasi et al. [1] proposed applying a recurrent neural network (RNN) for session-based recommendation and made significant progress. Then, Tan et al. [6] further improved the RNN model by taking data augmentation and distribution into consideration. RNN has natural advantages in modeling user sequence actions. It has proven very useful to capture a user’s general interests from a sequence. However, in terms of saving users’ long-term behavior information, RNN is not powerful enough, and it models only single-way transitions between session items. Recent years have witnessed significant developments in graph neural networks (GNNs) [7], which have proven useful when learning meaningful representations for graph data. The core idea of GNN is aggregating feature information from local graph nodes using...
neural networks. Hence, GNN naturally integrates the node information and has been demonstrated to be influential in representation learning [8], [9]. For session-based recommendations, Wu et al. [10] applied GNN to capture complex transitions of items and generate accurate item embedding vectors by modeling the item sequences of a session as a session graph. Based on actual item embedding, GNN learns reliable session representations and predicts the next-click item more precisely.

Although all of the methods mentioned above have made great progress in session-based recommendations, they still have some limitations. None of them addresses the so-called repeat behavior patterns, which makes up a large proportion of session-based recommendations. For example, people may repeatedly browse product information several times when shopping online, and we tend to listen to the same music frequently in a day [11]. Therefore, Ren et al. [12] proposed a model called RepeatNet, which incorporates a repeat-exploration mechanism into RNN to solve the problem of repeat consumption for session-based recommendations. However, RepeatNet uses GRU to learn the session representation, which is not powerful enough to describe the complex connections of session items.

To adequately capture the complex transitions between session items and accurately model users’ repeat-exploration behavior patterns, we propose a novel method called ReGNN, which combines GNN and a repeat-exploration mechanism to improve the performance of recommender systems. In ReGNN, we process the item sequence that a user interacts with during a session as a session graph, and each item is encoded as graph-structured data. By this method, the repeat behaviors of a user are translated as the edge information of the session graph, which can be accurately learned by GNN to formulate the item representation. Then, we apply an attention mechanism to capture the user’s main purpose in a current session, which is combined as a unified session representation. Finally, we employ a repeat-exploration mechanism to consider the user’s behavior patterns in repeat mode and explore mode and make an accurate prediction. Extensive experiments conducted on two benchmark datasets demonstrate the effectiveness of our proposed model.

In summary, the main contributions of our work are listed as follows:

• To the best of our knowledge, this is the first effort that incorporates the user’s repeat behavior information into GNN to improve the performance of session-based recommendations.

• ReGNN applies GNN and the attention mechanism to model the complex transactions between items in a session and integrates with the repeat-exploration mechanism to adequately capture users' repeated and exploring behavior patterns.

• We conduct extensive experiments on two real-world datasets, which proves the effectiveness of our model and demonstrates that our model significantly outperforms other state-of-the-art methods.

II. RELATED WORK

A. TRADITIONAL METHODS

Conventional methods for session-based recommendation include matrix factorization (MF) [13] and Markov chains (MC) [4]. MF factorizes a user-item rating matrix into low-rank matrices; one represents the user latent factors, and the other represents the item latent factors. This method captures the user’s general interests based on the user’s whole interaction history but ignores the information of the whole interactive sequence. Then, sequential methods based on MC were proposed, which predict a user’s next behavior based on the previous behavior. Yap et al. [14] introduced a competence score measure in personalized sequential pattern mining for next-item recommendations. However, the limitation is that it is difficult to capture users’ long-term preferences, and the state space becomes unmanageable when including all possible sequence selections [15].

B. DEEP LEARNING-BASED METHODS

Recently, deep neural networks have achieved great success in modeling sequential data. Hidasi et al. [1] applied RNN to model session representation and made great improvements over traditional methods. It learns session representations directly from previous interactions in a session and predicts the next items. Later, Tan et al. [6] further developed an RNN model by using a data augmentation technique to improve the model performance. Afterward, Li et al. [15] proposed a model called NARM with an encoder-decoder architecture for the first time. They employed an attention mechanism on RNN to capture the user’s main purpose in the current session, which leads to a promising improvement. Liu et al. [2] proposed a short-term attention/memory priority model (STAMP) to effectively capture the long-term and short-term interests of a session with an attention mechanism.

Currently, GNN has been widely used to generate graph-structured data because of its inherent capacity of modeling complex relations between graph nodes. Inspired by word2vec, Perozzi et al. [16] proposed a DeepWalk algorithm to learn node embedding in a graph based on random walks. Following DeepWalk, Tang et al. [17] proposed a network embedding algorithm called LINE. As a modification of GNN, Li et al. [18] proposed a model BPTT to compute gradients by incorporating GRU and GNN. For session-based recommendations, Wu et al. [10] proposed a model called SR-GNN, which encodes item sequences as a session graph and incorporates graph models into representing session sequences.

C. REPEAT RECOMMENDATION

Repeat behavior is a common phenomenon in different domains, including web visitation, shopping online, and listening to music. The frequent appearance of the users’ repeat behaviors in many scenarios illustrate the importance of repeat recommendations. Many scholars have actively explored this aspect. For example, Ehrenberg [19] studied the repeat buying behavior of customers for consumer bands.
by a negative binomial distribution (NBD) model. Later, Anderson et al. [11] found that recency is a key factor with a cache-based method, which indicates that users tend to buy items consumed recently. More recently, Benson et al. [20] incorporated time factors, distance factors and quality factors into a holistic model to handle the problem of sequential repeat consumption. Afterward, Ren et al. [12] proposed a repeat-explore mechanism to calculate the score of each candidate item score in two different modes for session-based recommendations.

Most of the session-based recommendation models mentioned above either simply employ GNN to learn the session representation without considering the importance of repeat recommendation. Ren et al. [12] utilize RNN to model the item transitions of a session, which are not powerful enough to capture the complex connection between session items. In a session where has repeated items, the recurrent neural network can not model the relationship between repeated items. However, GNN transforms the repeat information into the edges and learns the item connection to formulate accurate item representations. In this paper, we process the item sequence in a session as a session graph, by which GNN can adequately learn the user’s repeat behavior. In addition, we apply a repeat-exploration mechanism to model the user’s repeat and explore behavior patterns, which will further improve the performance of ReGNN. To the best of our knowledge, this is the first attempt to incorporate users’ repeat behavior information into GNNs for more accurate session-based recommendations.

### III. THE PROPOSED METHOD

#### A. TASK DEFINITION

The task of session-based recommendation is predicting the next item that the user will interact with when just given the user’s access behavior in the current session (e.g., behavior in a day). Here, we give a formulation of the problem.

Given a user’s action session \( I_s = [i_1, i_2, \ldots, i_j, \ldots, i_n] \), where \( i_j \) is the item that the user has visited in a sequence. The goal of our method is to predict the next item that the user will visit in candidate items when given a sequence of the user’s previous actions in a session. For example, we predict \( i_{n+1} \) when given \( I_s = [i_1, i_2, \ldots, i_j, \ldots, i_n] \). Furthermore, we denote \( x_j \in \mathbb{R}^d \) as the initial item embedding of \( i_j \). As a result, the sequence of a session can also be represented as \( [x_1, x_2, \ldots, x_j, \ldots, x_n] \). We have demonstrated the notation and explanations in Table 1.

| Notation | Explanation |
|----------|-------------|
| \( I_s \) | a user’s action session |
| \( S_g, S_c \) | global and current session representations |
| \( S_{re} \) | the combination of \( S_g \) and \( S_c \) |
| \( S_r, S_e \) | session representations in repeat and explore mode |
| \( E \) | edge adjacency matrix of session graph |
| \( x_j^t \) | item representation at time \( t \) |
| \( P_r, P_e \) | probability of repeat and explore mode |
| \( \alpha \) | attentive weight matrix |
| \( W, b \) | weight matrix and bias |
| \( \hat{y}_j^r, \hat{y}_j^e \) | item scores in repeat and explore mode |

#### B. MODEL FRAMEWORK

The workflow of our model is illustrated in Figure 1. ReGNN is an end-to-end model, and it consists of four layers. In the GNN embedding layer, we first encode the sequence of items in a session as a session graph and initial the item(node) embedding with random normal initializer, each item is a graph node, and some of them have outgoing edges and incoming edges. Then, each graph node is learned by the graph neural network to update the latent vectors of all interacted items. In this paper, we process the item sequence in a session as a session graph, by which GNN can adequately learn the user’s repeat behavior. In addition, we apply a repeat-exploration mechanism to model the user’s repeat and explore behavior patterns, which will further improve the performance of ReGNN. To the best of our knowledge, this is the first attempt to incorporate users’ repeat behavior information into GNNs for more accurate session-based recommendations.
users’ behaviors into repeat patterns and explore patterns. Then, we employ a repeat-explore mechanism to learn repeat mode session representation \((S_r)\) and explore mode session representation \((S_e)\). Finally, in the prediction layer, we calculate the score of visited and unvisited candidate items with different mode representations to obtain the final score of all items.

C. GNN EMBEDDING LAYER
To obtain the latent vectors of graph nodes, we apply graph neural networks to update the initial item embedding. GNN is suitable for item representation learning because it can extract features of session graphs by capturing the transitions between graph nodes. The latent vectors of items are processed by GNN as follows:

\[
e_i' = E^T [x_{i1}^{t-1}, x_{i2}^{t-1}, \ldots, x_{in}^{t-1}] + b \quad (1)
\]

\[
z_j' = \sigma (W_e e_j + U_j x_j^{t-1}) \quad (2)
\]

\[
r_j' = \sigma (W_e e_j + U_j x_j^{t-1}) \quad (3)
\]

\[
\hat{x}_j' = \tanh (W e_j + U (r_j' \odot x_j^{t-1})) \quad (4)
\]

\[
x_j' = (1 - z_j') \odot x_j^{t-1} + z_j' \odot \hat{x}_j' \quad (5)
\]

where \([x_{i1}^{t-1}, x_{i2}^{t-1}, \ldots, x_{in}^{t-1}]\) is the sequence of node vectors with the activated state at time \(t - 1\). The initial state of all node vectors in session \(I_s\) is \([x_1, x_2, \ldots, x_n]\). \(W, W_2, W_r, U_2, U_r, U\) are the weight matrices, and \(b\) is the bias unit. In addition, \(\sigma (\cdot)\) and \(\tanh (\cdot)\) are two commonly used activation functions, which refer to a sigmoid function and a tanh function. In addition, \(\odot\) is an elementwise multiplication operator, \(z_j'\) is an update gate and \(r_j'\) is a reset gate. Finally, \(E \in R^{n \times 2n}\) is the edge adjacency matrix of the session graph, as shown in Figure 2. We define \(E\) as a concatenation of two matrices \(E^{in} \in R^{m \times n}\) and \(E^{out} \in R^{m \times n}\), which describe the incoming edges and outgoing edges, respectively. However, the difference from a traditional adjacency matrix is that we define the total edge weights of \(E^{in}\) and \(E^{out}\) to be 1. For example, in Figure 2, node \(i_2\) has two incoming edges, \(e_1\) and \(e_2\), and their weights in \(E^{in}\) are 1/2. \(E^T\) is the transposed matrix of \(E\).

The overflow of item vector transition is that for each session graph, \(e_i'\) captures the connection of all nodes with the state at time \(t - 1\), and then, \(z_j'\) and \(r_j'\) determine what information will be updated or reset at time \(t\). Finally, we update all node states and obtain the accurate item representation \(x_j'\) at time \(t\).

D. SESSION GENERATING LAYER
After learning all the item representations, we further apply an attention mechanism to capture the user’s main preference in the current session and formulate the session representation. We denote \(s_g\) as the global session embedding and \(s_c\) as the local session embedding. For \(s_c\), we denote \(s_c = x_n'\)

\[
E^{in} \quad E^{out}
\]

\[
1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4
\]

\[
0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\]

\[
1/2 \quad 0 \quad 0 \quad 1/2 \quad 0 \quad 0 \quad 1/2 \quad 0\]

\[
0 \quad 1 \quad 0 \quad 0 \quad 3 \quad 0 \quad 0 \quad 0\]

\[
0 \quad 1/2 \quad 1/2 \quad 0 \quad 4 \quad 0 \quad 1 \quad 0\]

![FIGURE 2. An example of session graphs and connection metrics.](image)

where \(x_n'\) is the last-click item vector at time \(t\). The final session vector and the probability of repeat mode \(P_r\) and explore mode \(P_e\) are computed as follows:

\[
\alpha_j^{re} = W_{re1} \sigma \left( W_{re2} x_j' + W_{re3} x_n' + b_{re} \right) \quad (6)
\]

\[
s_g = \sum_{j=1}^{n} \alpha_j^{re} x_j' \quad (7)
\]

\[
s_{re} = W_{re4} [s_g, s_c] \quad (8)
\]

\[
[P_r, P_e] = \text{softmax} (W_{re} s_{re}) \quad (9)
\]

where \(W_{re1} \in R^{d \times 1}\), \(W_{re2} \in R^{d \times d}\), \(W_{re3} \in R^{d \times 1}\), \(W_{re4} \in R^{d \times 2d}\), \(W_{re} \in R^{d \times 2d}\) are weight matrices, which control the weights and the latent shape of item embedding vectors. \(\alpha_j^{re}\) is the weight score of item \(i_j\) in session \(I_s\), which combines the influence of \(x_j'\) and \(x_n'\). In Eq. 7, we employ an attention mechanism to obtain global session representation \(s_g\), and then, we concatenate the global session vector \(s_g\) and local session vector \(s_{re}\) in Eq. 8 and process the matrix with a linear transformation to obtain the final session representation \(s_{re}\). Finally, we calculate the probability distribution of the repeat mode and explore the mode by projecting \(s_{re}\) into a two-dimensional hidden space and processing it with a softmax function.

E. REPEAT-EXPLORE MECHANISM LAYER
1) REPEAT MODE
To consider the user’s repeat behavior pattern, we renewably calculate the session representation in this mode. All parameters are used to learn the repeat behavior characteristics of the user. We calculate the score of items that the user has interacted with in the current session as follows:

\[
\alpha_j' = W_{r1} \sigma \left( W_{r2} x_j' + W_{r3} x_n' + b_r \right) \quad (10)
\]

\[
s_{gr} = \sum_{j=1}^{n} \alpha_j' x_j' \quad (11)
\]

\[
s_r = W_{r4} [s_{gr}, s_c] \quad (12)
\]

\[
\hat{y}_j = I (i_j \in I_s) s_r' x_j \quad (13)
\]
where \( W_{r1} \in \mathbb{R}^{d \times 1} \), \( W_{r2}, W_{r3} \in \mathbb{R}^{d \times d} \) and \( W_{r4} \in \mathbb{R}^{d \times 2d} \) are weight matrices. \( s_{ge} \) is the global session vector in repeat mode, and \( s_r \) is the final session representation, which describes the user’s repeat behavior feature in this mode. Finally, we calculate the item score \( \hat{y}_j \) for the items that the user has interacted with in the current session. \( I(\cdot) \) is an indicator function. In Eq. 13, the score of an unvisited item will be 0, and we will calculate them in explore mode.

### 2) EXPLORE MODE

In this mode, we consider the situation in which the user visits new items that have not existed in the current session. The computing process is similar to that in repeat mode, and we calculate the item scores in explore mode as follows:

\[
a^e_j = W_{e1}x^j + W_{e2}x^j + W_{e3}x^j + b_e
\]

\[
s_{ge} = \sum_{j=0}^{n} a^e_j x^j
\]

\[
s_{e} = W_{e4}[s_{ge}, s_{e}]
\]

\[
\hat{y}^e_j = I(\{ij \neq I\}) s_{e} x^j
\]

where \( W_{e1} \in \mathbb{R}^{d \times 1} \), \( W_{e2}, W_{e3} \in \mathbb{R}^{d \times d} \), and \( W_{e4} \in \mathbb{R}^{d \times 2d} \) are weight matrices. \( b_e \) is a bias term. In this mode, all parameters are employed to learn the user’s explore behavior characteristics. \( \hat{y}^e_j \) only calculates the scores for the items that have not appeared in the user’s current session, and the scores of previous interacted items will be 0.

### F. PREDICTION LAYER AND PARAMETER LEARNING

For recommendation, we combine the item scores from both repeat mode and explore mode. Then, we use a softmax function to normalize the additive item score.

\[
\hat{y}_j = \text{softmax}(P_r \hat{y}^r_j + P_e \hat{y}^e_j)
\]

where \( P_r \) and \( P_e \) are the possibility distributions of the two modes. We can also regard them as the weight coefficient of two scores. \( \hat{y}_j \) denotes the final item score of item \( j \) in the candidate set.

For the loss function, we compute the cross-entropy of the item prediction and ground truth in a session as follows:

\[
L (\hat{y}_j) = - \sum_{j=1}^{m} y_j \log (\hat{y}_j) + (1 - y_j) \log (1 - \hat{y}_j)
\]

where \( y_j \) denotes the ground truth item of a session. In our experiments, we process it as a one-hot encoding vector. For model optimization, we minimize the objective function \( L (\hat{y}_j) \) when training the model, and all parameters are learned in an end-to-end backpropagation algorithm.

### IV. EXPERIMENTS

In this section, we will first briefly introduce the datasets and compare the baseline methods and evaluation metrics used in our experiments. Then, we compare ReGNN with other state-of-the-art methods and analyze the importance of different parameters in our model.
session-parallel mini-batch training process and a ranking-based loss function.

- **NARM [15]**: This is an RNN-based state-of-the-art model to capture the user’s main purpose by the attention mechanism.

- **STAMP [2]**: STAMP applies an attention mechanism to capture users’ general interests and current interests.

- **RepeatNet [12]**: This RNN-based method uses a repeat-exploration mechanism to consider the possibility of user repeat behavior.

- **SR-GNN [10]**: This baseline transfers an item sequence to a structured graph and uses GNN to learn the session representation.

C. EVALUATION METRICS

We use Precision@20 and Mean Reciprocal Rank@20 to evaluate all compared methods, which are defined as follows.

- **Precision@20 (P)** is a widely used metric in the SRS area, which represents the proportion of correctly recommended items in the top 20 positions.

- **Mean Reciprocal Rank@20 (MRR)** is the average of reciprocal ranks of desired item. The higher items are ranked in the recommendation list, the higher MRR scores will be. MRR is a normalized score of range [0,1].

D. EXPERIMENTAL SETUP

We set the dimension of item vectors to 100 for all datasets, and the default batch size is also 100. The number of training epochs is 30. Furthermore, we initialize all weight matrices using a random uniform initializer with a standard deviation of 0.1. In addition, the initial learning rate is set to 0.01, and its decay rate is 0.1 every three epochs. All of the experiments are conducted on a GeForce GTX 1080Ti GPU. We have released our experimental code and a small Yoochoose sub-dataset on GitHub (https://github.com/Jmax302/ReGNN).

E. PERFORMANCE COMPARISONS

To demonstrate the performance of ReGNN, we compare it with other state-of-the-art methods. Our experimental results are illustrated in Table 4. We evaluate the performance of different methods in terms of P@20 and MRR@20. From our experimental results, we make the following observations.

First, the most traditional and the simplest algorithms (i.e., POP and S-POP) perform poorly, as they consider only the popularity of items without the user history and personalized customization. Second, the traditional methods Item-KNN and BPR-MF are also not competitive because they cannot learn the information of the user’s action sequence. However, Item-KNN performs better than FPMC based on Markov chains. This phenomenon indicates that although FPMC considers the action sequences of a user, it only takes transitions over successive items, which may be problematic in making accurate predictions. For large data, FPMC requires too much memory. That is why the result is missing in Table 4 on the Yoochoose 1/4 dataset.

F. FURTHER INVESTIGATION

1) IMPACT OF EMBEDDING DIMENSION

To determine the impact of the item latent dimension on ReGNN, we conduct experiments with different dimensions varying from 1 to 300. Because Yoochoose 1/64 and Yoochoose 1/4 are only two different versions of the same

| Method | Yoochoose 1/64 | Yoochoose 1/4 | Diginetica |
|--------|---------------|---------------|------------|
|        | P | MRR | P | MRR | P | MRR |
| POP    | 6.71 | 1.65 | 1.33 | 0.30 | 0.91 | 0.23 |
| S-POP  | 30.44 | 18.35 | 27.08 | 17.75 | 21.07 | 14.69 |
| Item-KNN | 51.60 | 21.81 | 52.31 | 21.70 | 28.35 | 9.45 |
| BPR-MF | 31.31 | 12.08 | 3.40 | 1.57 | 15.19 | 8.63 |
| FPMC   | 45.62 | 15.01 | - | - | 31.55 | 8.92 |
| GRU4REC | 60.64 | 22.89 | 59.53 | 22.60 | 43.82 | 15.46 |
| NARM   | 68.32 | 28.63 | 69.73 | 29.23 | 62.58 | 27.35 |
| STAMP  | 68.74 | 29.67 | 70.44 | 30.23 | 62.03 | 27.38 |
| RepeatNet | 70.02 | 30.15 | 70.71 | 31.03 | 62.08 | 27.17 |
| SR-GNN | 70.57 | 30.94 | 71.36 | 31.89 | 63.03 | 27.42 |
| ReGNN  | 73.22 | 34.90 | 74.58 | 36.65 | 66.71 | 32.51 |

Improve(%) | 3.76 | 12.80 | 4.51 | 14.93 | 5.84 | 18.56 |

Furthermore, all neural-based methods outperform the conventional algorithms. This proves the effectiveness of deep learning technology in recommendations. GRU4REC applies GRU to capture the interest drift of the user’s sequential behavior and achieves a great performance. NARM and STAMP employ the attention mechanism to RNN and make a great improvement. Since RepeatNet uses a repeat-explore mechanism to consider user repeat behavior individually, it performs better than other previous methods. This proves that the repeat behavior of users is an important factor for the next-item recommendation, especially on datasets with high repeat ratios. Different from RNN-based methods, SR-GNN uses GNN to model the connections between items and obtains a good result. This indicates that GNN is superior to RNN on representation learning when given complex connected data.

Finally, our proposed model ReGNN outperforms all the methods mentioned above. When compared with the best baseline SR-GNN, the relative performance improved by ReGNN is 3.76%, 4.51%, 5.84% on P@20 and 12.80%, 14.93%, 18.56% on MRR@20 on the three datasets. The significant experimental results indicate that both parts of the GNN and the repeat-exploration mechanism are important to improve the performance of session-based recommendations. ReGNN not only achieves a great improvement on the relatively high repeat ratio dataset (Yoochoose) but also performs well on Diginetica. This is because on low repeat ratio datasets, the exploration mode of the repeat-exploration mechanism contributes more, and GNN can still learn an accurate session representation.

VOLUME 8, 2020
dataset, we conduct our experiments on Yoochoose 1/64 and Diginetica. Their performance is illustrated in Figure 3. It is obvious that ReGNN performs better with increasing dimension size on both datasets. This is because a high dimension can reflect the potential measurement better. However, when the dimension size is larger than 100, the performance of ReGNN only slightly improves. That is why we set the default dimension size to 100, considering the balance of effectiveness and efficiency.

2) IMPACT OF GNN AND THE REPEAT-EXPLORE MECHANISM

To verify the effect of the different parts in ReGNN, we compare ReGNN with two baselines without considering GNN or the repeat-exploration mechanism. We conducted experiments on Yoochoose 1/64 and Diginetica and show their results in Figure 4.

- **ReGNN-g.** This model removes the GNN part from ReGNN and replaces it with an RNN to learn the session representation. The repeat-explore mode is reserved.

- **ReGNN-re.** This model removes the repeat-exploration mechanism and simply calculates the item scores with the session representation learned by GNN.

First, the results shown in Figure 4 indicate that ReGNN outperforms both variant models (i.e., ReGNN-g and ReGNN-re) on both datasets. That is, removing the GNN part or the repeat-exploration mechanism will lead to performance loss. Furthermore, ReGNN-g performs slightly worse than ReGNN-re on Yoochoose 1/64 but significantly worse than ReGNN-re on Diginetica. This is because the dataset Yoochoose 1/64 has a relatively higher repeat ratio than Diginetica does. This indicates that the impact of the repeat-exploration mechanism varies with the different repeat ratios of the datasets. The repeat-exploration mechanism can achieve a great performance with frequent repeat behavior of users. In addition, ReGNN-re is relatively stable on both datasets. This shows that GNN can learn the accurate session representation in most scenarios and contributes more to the performance of ReGNN, especially on low repeat ratio datasets. Finally, it is obvious that both parts of the ReGNN are important to improve its performance and make more precise recommendations.

V. CONCLUSION

In this paper, we first studied the problem of the user’s repeat-explore behavior patterns in session-based recommendations. Then, we proposed a novel model called ReGNN to obtain the exact representation of a session by adequately modeling the complex relationships between items with GNN and the attention mechanism. Furthermore, we incorporated
a repeat-exploration mechanism into ReGNN to model the user’s repeat and explore behavior features. Extensive experiments conducted on the two benchmark datasets demonstrate the superiority of ReGNN. In future work, we will incorporate more item features into ReGNN, such as images and item description text, which can describe user preference more accurately, to improve the performance of ReGNN.

REFERENCES

[1] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based recommendations with recurrent neural networks,” in Proc. 4th ICLR, 2016.

[2] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, “STAMP: Short-term Attention/Memory priority model for session-based recommendation,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2018, pp. 1831–1839.

[3] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, “Personalizing session-based recommendations with hierarchical recurrent neural networks,” in Proc. 11th ACM Conf. Recommender Syst., Aug. 2017, pp. 130–137.

[4] G. Shani, D. Heckerman, and R. I. Brafman, “An MDP-based recommender system,” J. Mach. Learn. Res., vol. 6, pp. 1265–1295, Sep. 2005.

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

[6] Y. K. Tan, X. Xu, and Y. Liu, “Improved recurrent neural networks for session-based recommendations,” in Proc. 1st Workshop Deep Learn. Recommender Syst. DLRS, 2016, pp. 17–22.

[7] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. Adv. NIPS, 2017, pp. 1024–1034.

[8] T. Derr, Y. Ma, and J. Tang, “Signed graph convolutional networks,” in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2018, pp. 929–934.

[9] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, arXiv:1609.02907. [Online]. Available: http://arxiv.org/abs/1609.02907

[10] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, “Session-based recommendation with graph neural networks,” in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, pp. 346–353.

[11] A. Anderson, R. Kumar, A. Tomkins, and S. Vassilvitskii, “The dynamics of repeat consumption,” in Proc. 23rd Int. Conf. World Wide Web - WWW, 2014, pp. 419–430.

[12] P. Ren, Z. Chen, J. Li, Z. Ren, J. Ma, and M. de Rijke, “RepeatNet: A repeat aware neural recommendation machine for session-based recommendation,” in Proc. AAAI, vol. 33, Jul. 2019, pp. 4806–4813.

[13] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” Computer, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[14] G.-E. Yap, X.-L. Li, and S. Y. Philip, “Effective next-items recommendation via personalized sequential pattern mining,” in Proc. DASFAA, in Lecture Notes in Computer Science, vol. 7239. Berlin, Germany: Springer, 2012, pp. 48–64.

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

[16] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining KDD, 2014, pp. 701–710.

[17] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “LINE: Large-scale information network embedding,” in Proc. 24th Int. Conf. World Wide Web WWW, 2015, pp. 1067–1077.

[18] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” 2015, arXiv:1511.05493. [Online]. Available: http://arxiv.org/abs/1511.05493

XUEFENG XIAN received the Ph.D. degree in computer science from Soochow University, in 2012. He was a Visiting Scholar with George Mason University, in 2014. He is currently a Professor with Suzhou Vocational University. His research has been supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions. He has published more than 30 articles in international conferences and journals. His current research interests include data mining, web data management, and recommender systems.

LIGANG FANG received the M.S. degree in remote sensing and GIS and the Ph.D. degree in environmental science from the Graduate University of Chinese Academy of Sciences, Beijing, in 2003 and 2007, respectively. Since August 2007, he has been working with the Department of Computer Engineering, Suzhou Vocational University. He was a Visiting Scholar with Florida State University, in 2017. He is currently a Professor with Suzhou Vocational University. He has published more than 50 articles in international conferences and journals. His research interests include environmental remote sensing, sensor networks, and environment modeling.

SHIMING SUN received the B.Eng. degree in computer technology from the Nanjing Institute of Technology, Nanjing, China, in 2003, and the M.S. degree in computer software engineering from Hohai University, Nanjing, in 2006. He is currently a Senior Engineer with the Department of Basic Research and Development, NARI Group Corporation (State Grid Electric Power Research Institute), Nanjing. His research interests include electric power automation systems, recommender systems, and big data analysis.