Hypergraph Neural Networks with Attention Mechanism for Session-based Recommendation

Meirong Ding\textsuperscript{1a}, Xiaokang Lin\textsuperscript{1b\ast}, Biqing Zeng\textsuperscript{1c}, Yuan Chai\textsuperscript{2d}

\textsuperscript{1}School of Software, South China Normal University, Foshan 528225, China
\textsuperscript{2}Manning Selvage & Lee Public Relations Consultancy Beijing Co., LTD. Shanghai Branch, Guangzhou 20042, China
\textsuperscript{a}362034935@qq.com, \textsuperscript{b}\ast 2020023849@m.scnu.edu.cn, \textsuperscript{c}zengbiqing0528@163.com, \textsuperscript{d}cyuanpro@163.com

Abstract. The session-based recommendation task is designed to predict the behavior of the current session at the next moment based on multiple anonymous sessions. Due to the lack of user information in the session, the traditional recommendation model cannot be used directly to model the interest of specific users. In this paper, a session recommendation model based on hypergraph neural networks and attention mechanism (HGNNA) is proposed. Firstly, the features of items are learned by constructing hypergraph neural networks, then the conversation information is aggregated by self-attention mechanism, and finally the information among similar sessions is aggregated by graph attention networks. The hypergraph neural networks can capture the correlation between items, the self-attention mechanism can show the interest of the current session, and the graph attention networks can find the interest pattern between similar sessions, so that the representation vector of the session includes the information of the items in the session, other items outside the session and other sessions. In the experiments on two datasets, Yoochoose1/4 and Diginetica, the recommendation effect of HGNNA is higher than that of other relevant methods, especially in the P@20, which is improved by 0.69 and 1.40 respectively.

1. Introduction

With the rapid development of the Internet, the volume of information that internet users receive every day has been exploding in growth, and such an overwhelming amount of information often results in high information redundancy. A recommendation system (RS) can filter and select the information that suits users' interests and meets individual needs, bringing efficiency and convenience to our daily use of the Internet. Thus recommendation systems have been applied widely, such as in e-commerce, hot news, movie and music recommendations. Thereupon, algorithms used in recommendation systems have drawn academic interest and have become a hot research topic in academia and industry.

The present algorithms used in recommendation systems are mainly based on the known information of the users, however under real circumstances, users' information is often unknown and this gives rise to a class of recommendation algorithms - session-based recommendation algorithms. Session-based recommendations tend to infer a user's next action from multiple ongoing sessions, for example to infer the next item a user would click on based on their current browsing records. A session-based recommendation task has the following features:

(1) Due to the nature of "anonymity", the amount of information that can be utilized by a recommendation algorithm is quite limited.
A session-based recommendation task is not entirely equivalent to sequential prediction tasks, such as natural language processing and regression prediction. The data in a session-based recommendation task are items with topological structures, and the items in the same session do not have a strict sequential order, such as [computer, mouse, keyboard], and the information about the items is disordered but related.

The interests of different conversations may have similar patterns but there are subtle differences among them. For example, if both sessions are browsing records of computer products, the information of the two sessions can reference to each other.

Due to the above mentioned features, traditional recommendation algorithms cannot make proper recommendations from the data that lack user information. Direct applications of Markov chain or recurrent neural networks may place strict requirements in temporal order on data, resulting in excessive learning about time-dependent data, and additionally causing the learning of topological structure information in recurrent neural networks inadequate.

To perform a recommendation task more precisely by making the most use of the given information, Hypergraph Neural Networks with Attention Mechanism for Session-based Recommendation (HGNNA) is hereby proposed. Firstly, this model captures a correlation between items by constructing a hypergraph neural network, acquiring the topological structure information of the entire recommendation system, meanwhile it learns the importance of each item in the session. Secondly, the model aggregates session information through the self-attention mechanism to avoid strong temporal dependency, and showcases where the interest of the current session lies in. Finally, graph attention networks are used to transfer interest patterns between sessions, in order to better utilize limited information for recommendation and to prove better recommendation effects through experiments with publicly available datasets.

2. Related Works

2.1. Traditional recommendations
Traditional recommendation methods are mainly based on matrix factorization (MF), Item-KNN and Markov Chains (MC). Matrix factorization [1] represents the implied features of users and items by matrix factorization. However, session-based recommendation scenarios lack user information, which makes traditional matrix factorization unable to be used. The Item-KNN [2] calculates the similarity of sessions by the co-occurring items, but this method ignores the importance of different items in the current session. The famous traditional temporal model Markov Chain [3] predicts the next behavior by predicting the previous behaviors of the user, but such strict temporal dependency is not ideal for the recommendation tasks.

2.2. Recommended methods based on deep learning
As the applications of deep learning in solving practical problems have achieved good performance, deep learning also began to be used in recommendation tasks. Inspired by recurrent neural networks [4] and gated recurrent units [5], Hidasi et al. [6] proposed GRU4REC, which applies recurrent neural networks to predict users' sequential behaviours. Li et al. [7] proposed NARM that captures the importance of each item in the session by using an encoder-decoder model, recognizing and learning the intention of the whole session. Liu et al. [8] used the attention mechanism and multilayer perceptrons to capture users' interest in the past and the present, and thus proposed STAMP. However, these deep learning-based recommendation methods often disregard inter-item and inter-session information, leading to unsatisfactory performance in recommendations.

2.3. Recommendation methods based on graph neural networks
With the rise of graph neural networks (GNN) in recent years, many new forms of GNN have appeared [9-13]. Graph neural networks have been applied to natural language processing, image processing, social networks, and other fields [14], in the meantime, many graph neural networks have also been
applied to recommendation systems [15-17]. For session-based recommendation systems, SR-GNN proposed by Wu et al. [18] employs adjacency matrices and gated recurrent units to attain vectors of the current session, and then it combines all vectors of items with the attention mechanism to obtain the final session vector. Based on SR-GNN, Xu et al. [19] proposed GC-SAN which combines self-attention mechanism to study the proportional relationships among items and to connect them. Qiu et al. [20] constructed adjacency matrices to obtain the adjacency nodes with the depth of 2, and proposed FGNN model for node learning of graphs. Chen et al [24] used gated recurrent units to compensate for the temporal loss of graph information by connecting non-directly connected nodes, thus proposing the LESSR model. Graph neural networks can capture information among items, making them well suited for recommendation tasks with topological structures.

Recently, Feng et al. [22] proposed hypergraph neural networks, which are able to model sequential data more accurately than traditional graph neural networks. Each hyperedge can be constructed by any number of nodes instead of only two nodes, particularly when dealing with complex data, hypergraphs show more flexibility in data modeling. Therefore, hypergraph neural networks began to apply in recommendation systems. Xia et al. [23] proposed the DHCN model in which they constructed sessions in the form of a hypergraph, and used hypergraph convolutional neural networks to learn features among sessions. Wang et al. [24] designed hypergraph attention networks in spatial domains and aggregated session information with attention mechanism. However, under the circumstances of session-based recommendations, it is difficult for hypergraph neural networks to directly learn the interest patterns among different sessions.

3. Hypergraph Neural Networks with Attention Mechanism for Session-based Recommendation

In this section, we introduced HGNNA with the process shown in Figure 1. We first described the task, then proceeded to the description of hypergraph neural networks, self-attention layers and graph neural network modules, and finally to explain how to predict.

3.1. Task description

The session-based recommendation is according to the conversation set \( S = \{s_1, s_2, s_3, \ldots, s_M\} \), given the current session sequence \( s_i = [v_{i,1}, v_{i,2}, v_{i,3}, \ldots, v_{i,n}] \), and to predict the next most likely \( v_{i,n} \), which \( v_{i,j} \epsilon V, V = \{v_1, v_2, v_3, \ldots, v_N\} \) is a collection of all items, \( M \) is the size of the collection of the sessions, \( N \) is the size of the collection of items. The final output of the model is \( \hat{y} = (\hat{y}_1, \hat{y}_2, \hat{y}_3, \ldots, \hat{y}_N) \), with each dimension of \( \hat{y} \) representing a predicted probability of each item.

3.2. Modules of the model

3.2.1. Hypergraph neural networks. A hypergraph is defined as \( G = (V,E,W) \), \( V \) is the set of all nodes, \( E \) is the set of all hyperedges, and \( W \) is the weight matrix about the hyperedges. The hypergraph can be represented by the incidence matrix \( H \) of \( \mid V \times \mid E \), where \( v \epsilon V, h(v,e) = 1, \) where \( v \epsilon V \) and \( e \epsilon E \). The two matrices \( D_v \) and \( D_e \) represent the degree of nodes and the degree of hyperedges, respectively, where \( d(v) = \sum_{e \epsilon E} \omega(e)h(v,e) \) and \( d(e) = \sum_{v \epsilon V} h(v,e) \). The convolution of the hypergraph [22] is defined as:

![Figure 1. Hypergraph neural networks with attention mechanism for session-based recommendations](image-url)
where $X^{(l+1)}$ is the feature vectors of the nodes at $l$ layer, and $\theta^{(l)}$ is the weight matrix at $l$ layer. When $l = 0$, $X^{(0)}$ is the input vectors of the nodes.

In a spatial domain, the process of hypergraph convolutional neural networks can be viewed as two steps:

1. The information of nodes is aggregated into the information of hyperedges.
2. The information of hyperedges is aggregated into the information of nodes.

The information transfer process of hypergraph convolutional neural networks in the spatial domain is shown in Figure 2.

![Figure 2. Hypergraph convolutional networks](image)

A node on the hypergraph can be regarded as an item in the recommendation task, and through one iteration of the hypergraph convolutional neural network, information of the items aggregate with information of other items in first-order sessions.

### 3.2.2. Self-attention layers

Self-attention was proposed in the Transformer [25] and has been widely applied [19]. It aggregates and represents targets with different weights (similarity) by calculating the similarities between the target and other items in the sequence. The self-attention is defined as:

$$z = \text{softmax}(\frac{(xW^Q)(xW^K)\tau}{\sqrt{d}})(xW^V)$$

where $z \in \mathbb{R}^{m \times d}$ is the feature vectors after self-attention learning, $x = [x_1, \cdots]^T$, $x_i$ is the feature vector of the item contained in the session, $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$ are the feature transformation matrices, and $d$ is the dimension of the vector.

The self-attention layers drew on the idea of encoding layers in Transformer, and added positional encoders before a self-attention computation, and the self-attention computed feature vectors are stitched, linearly transformed, normalized, and residual concatenated, etc. We defined the self-attention layer as follows:

$$x' = SAN(x)$$

where $x'$ is the feature vectors after the whole self-attention layer learning and the whole self-attention layer is shown in Figure 3.
3.2.3. **Graph attention networks.** A graph attention network [10] learns feature vectors of each node through a graph structure. The nodes in the graph learn each other's attention coefficients and iteratively aggregate and update the vectors with the purpose of obtaining information of the neighboring nodes, and the learning process is shown in Figure 4.

First, the learning of attention coefficients $\alpha_{ij}$ between nodes in the graph is performed according to Eq. (4), and then, the feature information of neighboring nodes is aggregated into the central node feature $q_i \in \mathbb{R}^d$ according to Eq. (5). Finally at each iteration, the node aggregates the information of its first-order neighboring nodes.

$$\alpha_{ij} = \frac{\exp \left( \text{LeakyReLU}(\alpha^T [W^a p_i] || W^a p_j) \right)}{\sum_{k \in N_i} \exp \left( \text{LeakyReLU}(\alpha^T [W^a p_i] || W^a p_k) \right)} \quad (4)$$

$$q_i = \sigma(\sum_{k \in N_i} \alpha_{ij} W^a p_i) \quad (5)$$

where $\alpha^T \in \mathbb{R}^{2 \times d}$ is similarity calculation vectors, $W^a$ is feature transformation matrices, $N_i$ is the set of the neighboring nodes, and $\sigma$ is the activation function.
3.3. Our model: HGNNA

In this paper, we applied hypergraph neural networks to learn the features of each item, and used self-attention layers to aggregate the information of each item in a session. Due to the difficulty of describing interests and the difficulty of explaining neural networks, it is challenging for neural networks to learn the interest patterns of current sessions directly, therefore, we employed graph attention networks to aggregate similar sessions information to learn interest patterns of sessions. Given a set of sessions \( S = \{s_1, s_2, s_3, \ldots, s_M\} \), and a set of items \( V = \{v_1, v_2, v_3, \ldots, v_N\} \). The overall framework of the model is shown in Figure 1.

First, initialize the feature vector \( x_i \) for each item \( v_i \). Next, form the set of items \( V \) into a hypergraph, and the items in the same session \( s_i \) are connected by a hyperedge \( e_i \), i.e., a hyperedge represents a session. According to the definition of hypergraph, the incidence matrix \( H \) is constructed, and the degree \( D_v \) of nodes and the degree \( D_e \) of hyperedges can be calculated. Assuming that the feature matrix of items is \( X(0) \in \mathbb{R}^{N \times d} \), \( d \) is the feature dimension and the feature weight matrix \( \Theta(0) \) of each layer, \( X(1) \) can be learned using Equation (1). The number of iterative layers \( l \) chosen in this paper based on the experience of previous work related to graph neural networks [9] is 2.

Then, the item vectors \( x = [x_i, \ldots]^{T} \in \mathbb{R}^{m \times d} \) from \( X(1) \) learned by hypergraph neural networks in session \( s_i \) is selected as the input for learning in self-attention layers according to Equation (3). Take the last row vector \( x_m \) of the output item vector \( x' \in \mathbb{R}^{m \times d} \) as the feature vector \( p_i \) of session \( s_i \).

Next, the set of sessions \( S \) is formed into a graph. In this case, a node in the graph represents a session, and the existence of a connection between two nodes indicates that there is an intersection of the items in the two sessions. The feature vector \( p_i \) of each session \( s_i \) is used as the input for the graph attention network to learn the updated feature vector \( q_i \) of each session \( s_i \). In this paper, the number of iterations is chosen to be 2, based on previous work on graph neural networks [9].

Finally, the recommendation score \( \hat{y} \) is calculated based on Equation (6) and the feature vector of each candidate item, each dimension of \( \hat{y} \) can be regarded as the probability that each item is next clicked by session \( s_i \).

\[
\hat{y} = \text{softmax}(q_i X(l)^T)
\]  

The loss function \( \text{Loss}(\hat{y}) \) is defined as the cross-entropy function of the recommendation score and the true value, an it can be computed as follows:

\[
\text{Loss}(\hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log (1 - \hat{y}_i)
\]  

where \( y_i \in (0,1) \) indicates whether the next item clicked in the real session is \( v_i \), and \( \hat{y}_i \) is the recommendation score of candidate item \( i \) in \( \hat{y} \).

4. Experiments and Discussions

In this section, we first described the datasets, baseline methods, and evaluation metrics in the experiments. Then, we compared our model with other baseline methods and analyzed the results. Finally, we explored the importance of modules and the influence of hyper-parameters.

4.1. Datasets

In this paper, we evaluated our model on two datasets, Yoochoose1/4 and Diginetica. Yoochoose1/4 is the dataset from the RecSys Challenge 2015, containing users' browsing clicks on e-commerce pages. Diginetica is the dataset from the 2016 CIKM Cup, in which only transaction information data were used. The data is processed according to the paper [18]. i.e., items with more than or equal to 5 occurrences and sessions with more than or equal to 2 items are selected out, while duplicate items in a session are removed. Assuming a session \( s_i = [v_{i,1}, v_{i,2}, v_{i,3}, \ldots, v_{i,N}] \), multiple samples and labels of
samples can be produced $([v_{i,1}, v_{i,2}], [v_{i,1}, v_{i,2}, v_{i,3}], \ldots, [v_{i,1}, v_{i,2}, v_{i,3}, \ldots, v_{i,n-1}], v_{i,n})$. The information statistics of the datasets are shown in Table 1.

| Datasets       | Yoochoose1/4  | Diginetica |
|----------------|--------------|------------|
| Train          | 5,917,745    | 719,470    |
| Test           | 55,898       | 60,858     |
| Items          | 29,618       | 43,097     |
| Avg len        | 5.71         | 5.12       |

4.2. Baseline methods
To evaluate the effectiveness of the model, a comparison of the effectiveness of the baseline model was performed, and the baseline models are as follows:

1. The POP is a recommendation model based on the training set and the frequency of items in the current session.

2. Item-KNN [2] is a recommendation model that calculates the cosine similarity based on the items clicked in the session and the candidate items.

3. GRU4REC [4] uses RNN for training and prediction of session sequences.

4. NARM [7] uses RNN combined with attention mechanism for modelling and predicting sequence behaviours.

5. STAMP [8] performs next item prediction by capturing the general and current interests of users.

6. SR-GNN [18] uses graph neural networks for representations of implicit vectors and aggregations of information using traditional attention mechanism.

4.3. Evaluation Metrics
The following metrics are used to evaluate the performance of all models.

P@20 (Precision) represents the majority of correctly recommended items among the top 20 recommended items, which is defined as:

$$P@20 = \frac{n_{hit}}{n_r}$$

where $n_{hit}$ denotes the number of correct items among the top 20 recommended items, and $n_r$ denotes the total number of items in the test set.

MRR@20 (Mean Reciprocal Rank) is the sum of the inverse of the number of correctly recommended items. It represents the ranking of the top 20 recommended items in the range, and it is defined as:

$$MRR@20 = \frac{1}{n_r} \sum_{v \in V_{hit}} \frac{1}{rank_v}$$

Where, $V_{hit}$ denotes the set of correct items among the top 20 recommended items, $rank_v$ denotes the rank of item $v$ in the recommendation list, and $n_r$ denotes the total number of items in the test set.

4.4. Hyper-parameters settings
In this paper, different settings of several parameters were performed, including vector dimension $d = 100$, batch size $= 100$, iteration round $epoch = 40$ for Yoochoose1/4 dataset, iteration round $epoch = 30$ for Diginetica dataset, learning rate $lr = 0.001$, and $L_2$ regularization is $10^{-5}$. The parameters were optimized using the Adam optimizer.
4.5. Performance comparisons

In the experimental results in Table 2, the results in bold face are the best results among the same indicators, and the underlined results are the second best results among the same indicators. For the baseline model, since the datasets are the same, we used the results in the paper [18].

| models | Yoochoose1/4 | Diginetica |
|--------|--------------|------------|
|        | P@20 | MRR@20 | P@20 | MRR@20 |
| POP    | 1.33  | 0.30    | 0.89  | 0.20   |
| Item-KNN | 52.31 | 21.70   | 43.097| 11.57  |
| GRU4REC | 59.53 | 22.60   | 39.45 | 8.33   |
| NARM   | 69.73 | 39.23   | 49.70 | 16.17  |
| STAMP  | 70.44 | 30.00   | 45.64 | 14.32  |
| SR-GNN | 71.36 | 30.81   | 50.73 | 17.59  |
| HGNNA  | 72.05 | 31.89   | 52.13 | 17.74  |

From Table 2, the results of both traditional recommendation models and deep learning recommendation models are less than satisfactory, and neither of them made good use of the topology structure of the items. We can see that the proposed HGNNA performed better on both datasets. On Yoochoose1/4 dataset, HGNNA improved 0.69% over SR-GNN on P@20 and decreased 1.08% on MRR@20, but is still 0.81% higher than STAMP. On Diginetica dataset, HGNNA improved by 1.40% on P@20 and 0.59% on MRR@20 compared to SR-GNN. Considering the different meanings of the two metrics, it can be inferred that HGNNA is more effective for the recall of recommendation results, but less effective in the ranking of recommendation results. At the same time, the original feature vector of the last clicked item is very important for Yoochoose1/4 dataset [26], and the SR-GNN model with the original feature vector of the last clicked item showed better performance.

4.6. Ablation study

In this paper, we applied self-attention layers to aggregate session information and graph attention networks to capture information between sessions, both utilize attention mechanism. To verify the effectiveness of the attention mechanism in this model, ablation experiment was designed with the application of Diginetica dataset under the same conditions. HGNN-A is the model that aggregates the vector of each item on average as the session vector and removes graph attention networks. HGNNA-ATT is the model that is similar to HGNNA but without self-attention layers, that is, only aggregates the vector of each item on average as the session vector. HGNNA-GAT is a model similar to HGNNA but without graph attention networks. The HGNNA model is the standard model proposed in this paper.

| Model    | Diginetica |
|----------|------------|
|          | P@20 | MRR@20 |
| HGNN-A   | 50.50 | 17.25  |
| HGNNA-ATT| 50.58 | 17.27  |
| HGNNA-GAT| 51.87 | 17.52  |
| HGNNA   | 52.13 | 17.74  |

From Table 3, we can see that both HGNNA-ATT and HGNNA-GAT outperformed the HGNN-A model in P@20 and MRR@20 metrics, and both self-attention mechanism and graph attention networks can improve the model's effect respectively. Also, it can be deduced from the fact that HGNNA-GAT received better results than HGNNA-ATT, that self-attention mechanism improves the model more effectively than graph attention networks did. HGNNA reached the best result in both P@20 and MRR@20 metrics, indicating that a combination of the attention mechanism can improve the model's effect.
4.7. Hyper-parameters study

In this paper, the ablation experiments with self-attention layers $K = \{1,2,3,4\}$ were designed and completed for Yoochoose1/4 dataset and Diginetica dataset, and the experimental results are shown in Figure 5. The first 15 epochs of the training process for Diginetica dataset with different $K$ are shown in Figure 6.

According to Figure 5, on both Yoochose1/4 and Diginetica datasets, the best results were achieved at $P_{@20}$ when $K = 2$ and at $MRR_{@20}$ when $K = 3$. In this paper, we found that for $P_{@20}$ metric, the best results can be achieved with fewer layers of self-attention, while more layers may easily lead to a decrease in the results. For the $MRR_{@20}$ metric, more layers of self-attention than $P_{@20}$ are required in order to achieve the best result. Since the $MRR_{@20}$ metric includes the requirement of item ordering, using more layers (parameters) helps to improve the accuracy of the item ordering.

It can be observed from Figure 6 that for both the $P_{@20}$ and $MRR_{@20}$ metrics, the larger the number of layers $K$, the more epochs are required to reach the best result. For the same number of layers $K$, the $MRR_{@20}$ metric requires more epochs than the $P_{@20}$ metric to achieve the best results.

Combining Figure 5 and Figure 6, it can be inferred that the larger the number of parameters of the HGNNA model, the slower the $P_{@20}$ and $MRR_{@20}$ metrics reach the optimum.
5. Conclusion
To address the problems in session-based recommendations, this paper first uses hypergraph neural networks to learn information between items, then applies the self-attention mechanism to aggregate session information, and finally employs graph attention networks to learn about the information between sessions, thus proposing the HGNNA model. The experiments illustrate that the proposed model shows better effect on other advanced models, and the effect of each module is demonstrated by ablation experiments.

The proposed model showed different performance in different datasets, yet the characteristics of different datasets were not studied thoroughly in this paper. Meanwhile, the attention distribution of the weight matrix and incidence matrix of hypergraph convolutional neural networks were not considered. In future research, we will explore the above aspects.

Acknowledgments
This work was financially supported by the Project of Basic and Applied Basic Research Fund of Guangdong Province (2021A151501117). Authors wish to acknowledge the assistance from colleagues and the school of software.

References
[1] Koren Y, Bell R, Volinsky C. (2009) Matrix factorization techniques for recommender systems. In: Computer, 42(3): 30-37.
[2] Sarwar, Badrul, et al. (2001) Item-based collaborative filtering recommendation algorithms. In: The World Wide Web Conference. Hong Kong, China. 285-295.
[3] Rendle S, Freudenthaler C, Schmidt-Thieme L. (2010) Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on World wide web. North Carolina, USA. 811-820.
[4] Mikolov T, Karafiát M, Burget L, et al. (2010) Recurrent neural network based language model. In: Eleventh annual conference of the international speech communication association.
[5] Chang J, Gulcehre C, Cho K H, et al. (2014) In: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.
[6] Hidasi B, Karatzoglou A, Baltrunas L, et al. (2016) Session-based recommendations with recurrent neural networks. In: Proc of the 4thInt Conf of on Learning Representations. SanJuan, USA: ICLR.
[7] Li J, Ren P, Chen Z, et al. (2017) Neural attentive session-based recommendation. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. Singapore. 1419-1428.
[8] Liu Q, Zeng Y, Mokhosi R, et al. (2018) 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. 1831-1839.
[9] Kipf T N, Welling M. (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
[10] Veličković P, Cucurull G, Casanova A, et al. (2017) In: Graph attention networks. arXiv preprint arXiv:1710.10903.
[11] Wang X, Ji H, Shi C, et al. (2019) Heterogeneous graph attention network. In: The World Wide Web Conference. San Francisco, USA. 2022-2032.
[12] Yun S, Jeong M, Kim R, et al. (2019) Graph transformer networks. In: Advances in Neural Information Processing Systems. Vancouver, Canada. 32: 11983-11993.
[13] Hu Z, Dong Y, Wang K, et al. (2020) Heterogeneous graph transformer. In: Proceedings of The Web Conference 2020. Taiwan China. 2704-2710.
[14] Fan W, Ma Y, Li Q, et al. (2019) Graph neural networks for social recommendation. In: The World Wide Web Conference. San Francisco, USA. 417-426.
[15] Wu Q, Zhang H, Gao X, et al. (2019) Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In: The World Wide Web Conference. San Francisco, USA. 2091-2102.

[16] Zhang M, Wu S, Gao M, et al. (2020) Personalized graph neural networks with attention mechanism for session-aware recommendation. In: IEEE Transactions on Knowledge and Data Engineering.

[17] Ma C, Ma L, Zhang Y, et al. (2020) Memory Augmented Graph Neural Networks for Sequential Recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence. New York, USA. 34(04): 5045-5052.

[18] Wu S, Tang Y, Zhu Y, et al. (2019) Session-based recommendation with graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. California, USA. 33(01): 346-353.

[19] Xu C, Zhao P, Liu Y, et al. (2019) Graph Contextualized Self-Attention Network for Session-based Recommendation. In: IJCAI. Macao, China. 19: 3940-3946.

[20] Qiu R, Li J, Huang Z, et al. (2019) 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. 579-588.

[21] Chen T, Wong R C W. (2020) Handling Information Loss of Graph Neural Networks for Session-based Recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. USA. 1172-1180.

[22] Feng Y, You H, Zhang Z, et al. (2019) Hypergraph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. California, USA. 33(01): 3558-3565.

[23] Xia X, Yin H, Yu J, et al. (2020) Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation. arXiv preprint arXiv:2012.06852.

[24] Wang J, Ding K, Zhu Z, et al. (2021) Session-based Recommendation with Hypergraph Attention Networks. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM). Society for Industrial and Applied Mathematics. USA. 82-90.

[25] Vaswani A, Shazeer N, Parmar N, et al. (2017) Attention is all you need. Advances in neural information processing systems. 5998-6008.

[26] Yu F, Zhu Y, Liu Q, et al. (2020) TAGNN: Target attentive graph neural networks for session-based recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. China. 1921-1924.