A Dynamic Session-Based Recommendation System with Graph Neural Networks*

Vrushali Mahajan[1,*,†], Ermiyas Birihanu [1,†] and Tsegaye Misikir Tashu[2,†]

[1]ELTE Eötvös Loránd University, Department of Data Science and Engineering, Budapest, Hungary
[2]Department of Artificial Intelligence, Bernoulli Institute of Mathematics, Computer Science and Artificial Intelligence, University of Groningen, Groningen, The Netherlands

Abstract
Session-based recommendation systems provide personalized recommendations to users based on their session activities. Traditional recommendation algorithms often overlook temporal dependencies within user sessions, leading to suboptimal recommendations. To address this limitation, we proposed a Temporal Graph Neural Network (TemporalGNN) approach that leverages the temporal relationships between items in sessions to enhance recommendations. The proposed session-based recommendation system can effectively capture temporal dependencies within user sessions. The method consists of two temporal GNN layers designed to capture temporal dependencies within user sessions. A directed graph is constructed from session data, where each unique item in the dataset is a node, and directed edges are created between consecutive items within a session, with edge weights representing the time differences between interactions. This graph structure allows the model to capture the sequential nature of user interactions. The model generates recommendations by computing similarity scores from the learned embeddings and selecting the top N items with the highest scores. The experimental results on two real-world datasets showed the effectiveness of the proposed method in improving recommendation performance compared to the baseline approaches. The source code implementation is available on the GitHub repository at https://github.com/VrushaliM/SB-Recommendation-GNN.

Keywords
Session, User, Graph Neural Networks, Recommendation Systems.,

1. Introduction

In today's online world, personalized suggestions appear frequently while shopping, watching videos, or browsing the Internet. These suggestions are made by recommendation systems, which predict our preferences based on past behaviour. However, session-based recommendation systems have emerged since our tastes can change quickly during short online sessions. These systems focus on our current online activities to provide accurate suggestions for what we might like in real-time [1] [2]. Various types of recommendation systems exist in the research landscape, such as collaborative filtering, content-based filtering, and hybrid approaches [3] [4] [5]. Collaborative filtering generates recommendations based on similarities in user behaviour, while content-based filtering uses item characteristics. Hybrid approaches combine both methods for better accuracy. These studies consider collaborative filtering within sessions, capturing user-item interactions to handle sparse data and the cold-start problem effectively.

Researchers have been advancing these recommendation systems by exploring various techniques. These efforts include analyzing the sequential order of user interactions online and employing sophisticated models, such as neural networks [1] [6]. These studies investigated online sessions' duration, temporal dynamics, and underlying user intents. However, this approach may need to include other relevant information, potentially limiting recommendation diversity. Session-based recommendation focuses on predicting the next click of an anonymous user based on their current session activity [7]. Two popular models for session-based recommendations are Markov Chains (MC) and Recurrent Neural Networks (RNN) [8]. Graph Neural Networks (GNNs) have emerged as a promising approach. GNNs improve node representations by incorporating adjacent information through weighted or unweighted edges, making identifying item transitions easier [8]. To effectively explore long-range vertex relationships in a graph, a GNN typically requires three layers to propagate information. However, more than three layers can result in over-smoothing, where the distinctions between items are lost [9].

Despite progress, challenges remain, such as quickly identifying preferences with limited information and adapting to rapidly changing user behaviours. One of the most critical issues in session-based recommendation is how to accurately and efficiently capture and learn complex transitions of items from limited information. To ad-

ITAT 2024: Information Technologies – Applications and Theory, September 20–24, 2024, Drienica, Slovakia
*Corresponding author.
†These authors contributed equally.
y2hse8@inf.elte.hu (V. Mahajan); ermiyasbirihanu@inf.elte.hu (E. B.); t.m.tashu@rug.nl (T. M. Tashu)

dress these challenges, we propose a novel approach for session-based recommendation systems that can quickly understand user preferences, adapt to online behaviour, and efficiently handle large data volumes. This study focuses on time-driven session-based recommendation systems, integrating temporal order and time interval information to enhance accuracy and relevance. Our proposed method aims to improve time-driven session-based recommendations by considering the order of interactions and the specific duration between them. This approach is designed to uncover significant correlations between interacted items. Our goal is to enhance the accuracy and relevance of online recommendations, making them more helpful and enjoyable for users by exploring the impact of time intervals between interactions alongside the chronological order.

2. Related work

Session-based recommendation systems have garnered significant attention in recent years due to their ability to deliver personalized recommendations in real-time, reflecting users' immediate interests and preferences. This section provides an overview of the latest developments, methodologies, and findings in the field of session-based recommendation systems, highlighting key trends and areas of research focus [10] extends primary Recurrent Neural Network (RNN) models for session-based recommendation by incorporating data augmentation techniques and addressing shifts in input data distribution, leading to significant performance improvements over traditional approaches. In [11] introduced Session-based Recommendation with Graph Neural Networks (SR-GNN), utilizing graph-structured data to represent session sequences and capture item transitions, enhancing user representations. Zhang et al. [12] proposed A-PGNN (Attention-enhanced PGNN), which utilizes a graph neural network to capture complex item relationships in user behaviour graphs, integrating the DotAttention mechanism to model the impact of historical sessions on current sessions, facilitating long-term user preference capture.

The study in [13] developed a graph hierarchical dwell-time attention network to better capture user preferences and improve recommendation accuracy by incorporating a modified graph neural network and a hierarchical dwell-time attention module. The study in [14] proposed Attention-enhanced Graph Neural Networks with Global Context, modeling session-aware attention mechanisms and graph convolutional networks to learn and merge item transitions from all sessions. Yupu, G et al. [13] proposed a Time-Aware Graph Neural Network (TA-GNN), considering both long-term historical behaviours and collaborative filtering information from neighbouring users by constructing user behaviour and neighbourhood graphs and incorporating time interval information. Guojia An et al. [14] proposed a method that uses graph structures for hierarchical feature learning in item embeddings and automatically selects focus area lengths for session embeddings. Wang et al. [15] proposed a method that divides sessions into time slices and constructs temporal graphs and hypergraphs to capture item transitions over time in the session-based recommendation. However, there is a computational overhead associated with modeling item transitions across multiple time slices, which may affect scalability as the number of sessions and time slices increases

Zhu et al. [16] proposed the DGS-MGNN method, which dynamically constructs local, global, and consensus graphs to capture item representations from multiple perspectives, enhancing session-based recommendation accuracy. Sheng et al. [17] modeled interaction sequences using a Weighted Global Item Graph and current sessions with a Local Session Graph, integrating multiple interaction patterns. While significant advancements have been made in session-based recommendation systems, challenges remain, such as computational complexity, scalability, and the need for more comprehensive user preference modeling. This study proposes methods that address these challenges using dynamic session-based graphs, aiming to improve the accuracy and efficiency of recommendations in diverse and large-scale datasets.

Session-based recommendation systems aim to deliver personalized recommendations in real time based on users' immediate interests and preferences. Despite significant progress, challenges persist, including high computational complexity, scalability issues, and the need for more comprehensive modeling of user preferences. Hence, this study aims to improve time-driven session-based recommendations by considering the order of interactions and their specific duration, targeting to enhance the model's performance.

3. Proposed method

The proposed method, TemporalGNN, uses user session data to construct a graph where each session represents a sequence of interactions with items over time. Nodes in the graph are items, and edges reflect the order of interactions with time differences as edge features. The TemporalGraph Neural Network (TemporalGNN) consists of stacked TemporalGNNLayers that capture temporal dependencies and learn item representations. During training, the model updates item embeddings based on the temporal context of interactions, learning patterns and relationships between items. After training, recommendations are generated by calculating similarity scores from the learned embeddings and selecting the top-N items with the highest scores.

A temporal graph neural network models relationships and interactions between nodes while considering the timing of these interactions. The input is a graph with nodes representing items and edges representing sequential interactions with time differences. The output is a set of learned item embeddings that capture structural and temporal relationships, enabling personalized and context-aware recommendations. First, a directed temporal graph is constructed from the training data, with nodes representing unique items and edges encoding the temporal order of session interactions. The TemporalGNN model, with two TemporalGNNLayers, analyzes the temporal structure of the graph data. During training, the model updates node features by considering the temporal context and optimizing model parameters to minimize differences between positive items within the same session and different session interactions. The trained model makes recommendations by computing representations for items within a session and using their similarity to session items. The learned node features enhance the model's ability to make personalized recommendations by capturing temporal variations in item interactions. Algorithm 1 and Figure 1 show how the proposed method works.

Algorithm 1 Top-N Recommendation System

Input: D - Input data, G - Graph structure, d_{in} - Input dimension, d_{hidden} - Hidden layer dimension, d_{out} - Output dimension
Output: Top-N recommended items
Construct Graph
Create a directed graph $G = (V, E)$ from the input data D.
Define TemporalGNN Model
Initialize the TemporalGNN model with:
    Input layer of dimension d_{in}
    Hidden layer of dimension d_{hidden}
    Output layer of dimension d_{out}
Message Passing Through Layers
for each layer in the GNN model **do**
    Refine node embeddings using attention mechanism
    Capture fine-grained dependencies between items
end for
Train Model
Train the TemporalGNN model by minimizing the loss function using training data.
Compute Similarity Scores
Compute similarity scores between items using embeddings obtained from the trained model.
Recommend Items
Recommend the top-N items based on the computed similarity scores.

3.1. Graph construction

The interaction records D contain information about user sessions, items, and timestamps. Each session records a sequence of items interacted with by a user, along with the corresponding timestamps. A mapping M is created to associate each unique item i with a unique node ID n_i. This mapping translates item interactions into node interactions in the graph.

$$M : i \rightarrow n_i$$

The graph $G = (V, E)$ is constructed where:

- V is the set of nodes, each corresponding to a unique item.
- E is the set of directed edges representing temporal relationships between items within a session.

We identify the unique items and their corresponding node indices for each session using the mapping M. Directed edges are added between consecutive items in each session. The edges carry weights representing the time difference between consecutive interactions. If a session involves items {item1, item2, item3} with corresponding timestamps {t1, t2, t3}, the directed edges and their weights, that is, time differences are as follows:

$$e_{01} = (n_{\text{item1}}, n_{\text{item2}}), \text{ weight} = t2 - t1$$
$$e_{12} = (n_{\text{item2}}, n_{\text{item3}}), \text{ weight} = t3 - t2$$

For a session $S = [(i_1, t_1), (i_2, t_2), \ldots, (i_k, t_k)]$:

$$V = \{M(i_1), M(i_2), \ldots, M(i_k)\}$$
$$E = \{(M(i_j), M(i_{j+1})) \mid \forall j \in [1, k-1]\}$$
$$\text{Weight of edge}(M(i_j), M(i_{j+1})) = t_{j+1} - t_j$$

The graph is then transformed into matrix representations:

- **Adjacency Matrix** A: An adjacency matrix represents the connections between nodes. If there is a directed edge from node i to node j with a weight (time difference), A_{ij} is the weight; otherwise, $A_{ij} = 0$.

$$A_{ij} = \begin{cases} t_j - t_i & \text{if there is an edge from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

- **Feature Matrix** X: Each node v_i have associated features. We used an identity matrix where each node's feature vector is a one-hot encoded vector corresponding to the item.

$$X = I_{num\_items}$$

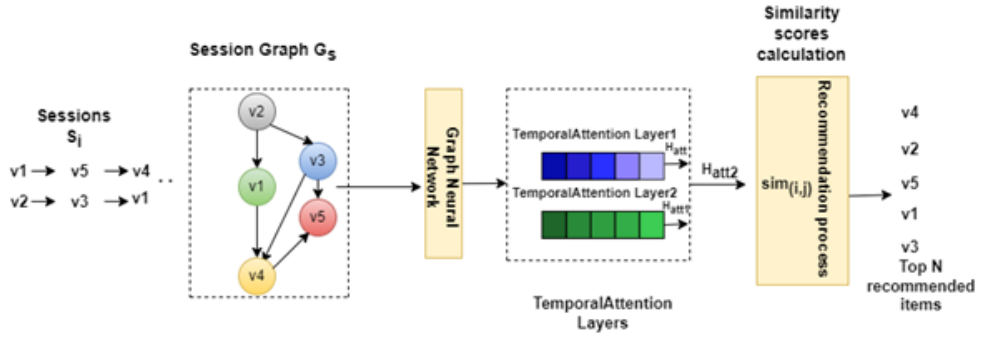

Figure 1: An architecture of proposed TemporalGNN model showing flow of data from Graph construction from session to item recommendation.

After constructing the graph G and obtaining the adjacency matrix A and feature matrix X, the data is fed into the Graph Neural Network. The adjacency matrix A captures the structure of the graph and the temporal relationships between items, while the feature matrix X provides the initial feature representation for each node. GNNs outperform RNNs when dealing with graph-structured data, as they are adept at capturing intricate relationships and processing non-Euclidean data. GNNs are capable of handling inputs of varying sizes and structures, making them particularly advantageous for tasks involving complex relationships and graph-structured data [18]. In contrast, while RNNs excel with sequential data, GNNs provide significant benefits for these more complex scenarios.

3.2. Temporal Graph Neural Network

The TemporalGNN model processes graph data and item-to-node mapping to extract meaningful representations of items within the graph. It consists of TemporalGNNLayers that update node features based on their temporal context within the graph. The TemporalGNN model is initialized with parameters where N is the number of nodes, H is the number of hidden units, and O is the number of output features.

The forward pass of the TemporalGNN model is performed through TemporalGNNLayers, which update node features iteratively. $F^{(0)}$ represent the initial node features. The forward pass of the TemporalGNN model can be expressed as:

$$F^{(l+1)} = \text{TemporalGNNLayer}(G, F^{(l)})$$

where l represents the layer index, $F^{(l)}$ is the node features at layer l, and $F^{(l+1)}$ is the updated node features after passing through layer l.

3.2.1. TemporalGNNLayer

The TemporalGNNLayer updates node features based on their temporal context within the graph. Let N_i denote the set of neighbouring nodes of node i.

The graph G and initial node features $F^{(l)}$ at layer l, the forward pass of the TemporalGNNLayer is:

$$F_i^{(l+1)} = \text{Update}(F_i^{(l)}, \{F_j^{(l)} \mid j \in N_i\})$$

where $F_i^{(l)}$ represents the feature vector of node i at layer l, and $\{F_j^{(l)} \mid j \in N_i\}$ denotes the set of feature vectors of neighboring nodes of i.

The TemporalGNN model consists of two TemporalGNNLayers. During the forward pass, each TemporalGNNLayer updates node features based on their temporal context within the graph. The first layer uses an attention mechanism to consider the importance of neighbouring nodes, while the second layer aggregates information from neighbouring nodes using mean pooling. The final node features generated by the TemporalGNN model encode meaningful representations of items within the temporal graph. These representations capture the temporal context of item interactions within sessions and can be seen as embeddings of items.

The TemporalGNN model is trained using a training dataset D_{train}, consisting of session data and corresponding item interactions. During training, the model learns to predict item sequences within sessions and refine its node representations.

The training process involves optimizing model parameters to minimize a loss function $\mathcal{L}$ defined over the training data:

$$\min_{\Theta} \sum_{(S_i, S_{\text{neg}})} \mathcal{L}(S_i, S_{\text{neg}})$$

Where Θ represents the model parameters, S_i denotes positive samples, i.e. items within the same session, and

S_{neg} denotes negative samples, i.e. items from different sessions. The model is trained using pairwise hinge loss to optimize the parameters so that the embeddings of items interacted with in the same session are closer to each other than to the embeddings of items not interacted with.

3.3. Recommendation process

The recommendation process involves the selection of top-N items for users based on their activities in the current session. The value of N is decoded as five (5) based on average lenghts of sessions in our dataset. The recommendation process uses a trained TemporalGNN model to generate item recommendations for sessions. The final node features obtained from the TemporalGNN model serve as input features for the recommendation task. When recommending items for a specific session, the method first identifies the items in the current session and retrieves their embeddings. It then computes the similarity scores between these session item embeddings and all other item embeddings. The method ranks all items based on these similarity scores and selects the top 5 items with the highest scores as recommendations. These items are considered the most similar or relevant to the current session's context.

3.3.1. Recommendation

Given a session S_q, the model computes representations for items within the session using its learned parameters. It then recommends items based on their similarity to items in the session, aiming to maximize the relevance of recommendations. This process can be expressed as:

$$\text{Recommend}(S_q) = \arg\max_{i \notin S_q} \text{Sim}(F_i, F_q)$$

where F_i represents the node features of item i, F_q represents the aggregated features of items in session S_q, and $\text{Sim}(\cdot)$ denotes a similarity measure between node features.

Similarity Score: The similarity between node features F_i and F_q are computed using Cosine similarity, and it is calculated as:

$$\text{Cosine Similarity}(F_i, F_q) = \frac{F_i \cdot F_q}{\|F_i\| \cdot \|F_q\|}$$

where $F_i \cdot F_q$ represents the dot product of the node feature vectors, and $\|F_i\|$ and $\|F_q\|$ represent their respective Euclidean norms. The embeddings of items within a session are obtained from the output of the temporalGNN model, which provides updated node features representing each item's embedding. Similarity scores between items are computed based on cosine similarity, which measures the cosine of the angle between two vectors, representing the similarity between their directions in the embedding space. This calculation is carried out by taking the dot product of the embeddings of items within a session and then dividing it by the product of their magnitudes, resulting in cosine similarity scores. Finally, items are ranked based on similarity scores, determining the order in which they are recommended to the user. This process enables the model to recommend items that are most similar to those already interacted with by the users. The method recommends the top 5 items by computing and ranking similarity scores for items within a specific session. The recommendation process is applied to each session individually to provide personalized recommendations to users.

4. Experiments and evaluation

4.1. Dataset and performance metrics

We evaluated our model on the Yoochoose [1] and Diginetica datasets [2] [15] [19]. The yoochoose dataset is obtained from the RecSys'15 Challenge. The dataset captures user-item interactions collected from an online retailer over a period of several months. It contains anonymized information about user sessions, including the items users viewed and purchased during each session, as well as timestamps indicating when these interactions occurred. Diginetica is a competition dataset used in CIKM Cup 2016. It contains user sessions extracted from an e-commerce search engine logs. The dataset includes columns such as session-id, user-id,item-id, eventdate and timeframe.

Table 1
Statistics of the Yoochoose and Diginetica datasets

Statistics	Yoochoose	Diginetica
# clicks	557,248	982,961
# train	369,859	719,470
# test	55,898	60,858
# items	16,766	43,097
avg length	6.16	5.12

We consider MRR@k and recall@k metrics to evaluate the performance of our model, and We set k to 5, which is suitable for session-based recommendations. Mean Reciprocal Rank measures how well a list of ranked items matches a set of true items. It is the average of the reciprocal ranks of the true items in the ranked list.

$$\text{MRR@k} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i}$$

[1] https://darel13712.github.io/rs_datasets/Datasets/yoochoose/
[2] https://darel13712.github.io/rs_datasets/Datasets/diginetica/

Where N is the number of sessions, and rank_i is the rank position of the true item in the i-th session's recommendation list.

Recall measures the proportion of true items successfully recommended in the top-k items. It is the ratio of the number of true items found in the top-k recommendations to the total number of true items. It is mathematically defined as:

$$\text{Recall@k} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{\text{true\_item}_i \in \text{top\_k\_recommended}_i\}$$

Where $\mathbf{1}\{\cdot\}$ is an indicator function that is one if the condition inside is true and 0 otherwise.

4.2. Baselines

To demonstrate the performance of the proposed method, we compared it with the contextual K-Nearest Neighbours approach (CKNN) [20] and graph-based Node2Vec approach [21] [22]. **CKNN** algorithm is a straightforward yet effective method for recommendation [20]. In the context of session-based recommendation, KNN recommends items most similar to the items in the current session based on historical co-occurrence data.

Node2Vec approach integrates network-based representation learning, clustering, and personalized recommendation techniques to provide practical and personalized recommendations for users in session-based scenarios.

4.3. Experimental settings

We used hyperparameter tuning and cross-validation to optimize the performance of the TemporalGNN model. The hyperparameters tuned include the number of hidden units in the first GNN layer, the number of output features in the second GNN layer, the learning rate for the Adam optimizer, and the number of training epochs. The search space for these parameters includes 32, 64 and 128 values for hidden units 16, 32 and 64 for output feature 0.001, 0.005 and 0.01 for learning rate and 10, 20 and 30 for a number of epochs. We used a random search strategy to explore this search space by sampling ten different sets of hyperparameter combinations. For each sampled combination, 5-fold cross-validation is conducted to evaluate the model's performance, which involves splitting the training data into five folds and iterating five times, each time using a different fold as the validation set. In comparison, the remaining four folds are used for training. During each fold, the model is trained with the specified hyperparameters, and its performance is evaluated using the Mean Reciprocal Rank metric, which measures the quality of the ranking of recommended items. The MRR scores from all five folds are averaged to obtain a single performance score for the hyperparameter set. The combination of hyperparameters that gives the highest average MRR across the five folds is considered the best. This optimal set of hyperparameters is then used to train the final model on the entire training dataset.

4.4. Experimental results

Figure 2 shows the performance of the TemporalGNN model on both datasets in terms of MRR@5 and Recall@5.

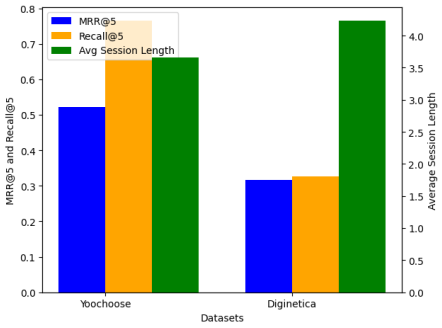

Figure 2: Performance of proposed model

On the Yoochoose dataset, TemporalGNN significantly outperforms the other models with an MRR@5 of 0.52 and a Recall@5 of 0.54. This indicates that TemporalGNN is highly effective in ranking the correct items near the top and retrieving relevant items within the top 5 recommendations. In contrast, KNN shows very poor performance with an MRR@5 of 0.069 and Recall@5 of 0.07, suggesting that it struggles to provide useful recommendations. Node2vec performs better than KNN with an MRR@5 of 0.13 and Recall@5 of 0.3, indicating moderate effectiveness. However, it still falls short of the performance achieved by TemporalGNN, highlighting the superiority of TemporalGNN's ability to utilize temporal and graph-based features for recommendation tasks.

Table 2
Performance of proposed model and baseline models

Model	Yoochoose		Diginetica	
	MRR	Recall	MRR	Recall
TemporalGNN	0.52	0.54	0.29	0.30
CKNN	0.06	0.07	0.02	0.02
Node2vec	0.13	0.30	0.14	0.50

On the Diginetica dataset, TemporalGNN's performance stands out, surpassing the other models with an MRR of 0.29 and Recall of 0.30. In comparison, KNN's

performance is notably lower, with an MRR of 0.022 and Recall of 0.02, and Node2vec, while better than KNN, still falls behind with an MRR of 0.14 and Recall of 0.5. These results underscore the superior performance of TemporalGNN in capturing the temporal dynamics of user interactions, leading to more accurate and relevant recommendations. Similarly, on the Diginetica dataset, TemporalGNN showed better performance with an MRR@5 of 0.29 and Recall@5 of 0.30. Although these scores are lower than their performance on Yoochoose, TemporalGNN remains the best model, indicating its robust capability across different datasets. KNN, on the other hand, performs very poorly with an MRR@5 of 0.013 and Recall@5 of 0.01, showing that it is not well-suited for this dataset. Node2vec shows a notable improvement in Recall@5 with a score of 0.5 but has an MRR@5 of 0.14, indicating that while it can retrieve relevant items better, it does not rank them as highly as TemporalGNN. This comparison further highlights the effectiveness of TemporalGNN in leveraging complex temporal and graph-based interactions for better recommendation performance.

5. Conclusion and future work

We introduced and evaluated a Temporal Graph Neural Network method for session-based recommendation tasks by showing its effectiveness on the Yoochoose and Diginetica datasets. We implemented the TempralGNN model to encode temporal dynamics within the graph, which extracted meaningful representations of items based on their temporal context to recommend top-N items to users. TempooralGNN outperformed traditional methods like CKNN and graph-based Node2Vec and achieved higher performance in terms of Mean Reciprocal Rank and Recall scores.

Future work could incorporate contextual information, such as the device used, location, or time of day, to provide more contextually relevant recommendations. There are exciting possibilities for future research in the field of session-based recommendation tasks. Enhancements to the TemporalGNN architecture, such as incorporating attention mechanisms and exploring alternative graph structures, could further improve its performance and scalability. We will include a comparative analysis of various approaches alongside Temporal GNN. This should pique the audience's interest in the potential for further research and development in the field.

References

[1] B. Hidasi, M. Quadrana, A. Karatzoglou, D. Tikk, Parallel recurrent neural network architectures for feature-rich session-based recommendations, in: Proceedings of the 10th ACM conference on recommender systems, 2016, pp. 241–248.

[2] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, X. Zhou, Graph contextualized self-attention network for session-based recommendation., in: IJCAI, volume 19, 2019, pp. 3940–3946.

[3] F. Yu, Y. Zhu, Q. Liu, S. Wu, L. Wang, T. Tan, 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, 2020, pp. 1921–1924.

[4] M. Zhang, S. Wu, M. Gao, X. Jiang, K. Xu, L. Wang, Personalized graph neural networks with attention mechanism for session-aware recommendation, IEEE Transactions on Knowledge and Data Engineering 34 (2020) 3946–3957.

[5] Y. K. Tan, X. Xu, Y. Liu, Improved recurrent neural networks for session-based recommendations, in: Proceedings of the 1st workshop on deep learning for recommender systems, 2016, pp. 17–22.

[6] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based recommendations with recurrent neural networks, arXiv preprint arXiv:1511.06939 (2015).

[7] C. Ding, Z. Zhao, C. Li, Y. Yu, Q. Zeng, Session-based recommendation with hypergraph convolutional networks and sequential information embeddings, Expert Systems with Applications 223 (2023) 119875.

[8] J. Wang, H. Xie, F. L. Wang, L.-K. Lee, M. Wei, Jointly modeling intra-and inter-session dependencies with graph neural networks for session-based recommendations, Information Processing & Management 60 (2023) 103209.

[9] R. Qiu, J. Li, Z. Huang, H. Yin, 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, 2019, pp. 579–588.

[10] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: Proceedings of the AAAI conference on artificial intelligence, volume 33, 2019, pp. 346–353.

[11] H. Rong, W. Zhu, C. Zhu, Graph hierarchical dwell-time attention network for session-based recommendation, in: ITM Web of Conferences, volume 47, EDP Sciences, 2022, p. 02032.

[12] Y. Chen, Y. Tang, Y. Yuan, Attention-enhanced graph neural networks with global context for session-based recommendation, IEEE Access 11 (2023) 26237–26246.

[13] Y. Guo, Y. Ling, H. Chen, A time-aware graph neural network for session-based recommendation, IEEE

Access 8 (2020) 167371–167382.
[14] G. An, J. Sun, Y. Yang, F. Sun, Enhancing collaborative information with contrastive learning for session-based recommendation, Information Processing & Management 61 (2024) 103738.
[15] H. Wang, S. Yan, C. Wu, L. Han, L. Zhou, Cross-view temporal graph contrastive learning for session-based recommendation, Knowledge-Based Systems 264 (2023) 110304.
[16] Z. Sheng, T. Zhang, Y. Zhang, S. Gao, Enhanced graph neural network for session-based recommendation, Expert Systems with Applications 213 (2023) 118887.
[17] F. Wang, X. Gao, Z. Chen, L. Lyu, Contrastive multi-level graph neural networks for session-based recommendation, IEEE Transactions on Multimedia 25 (2023) 9278–9289.
[18] I. R. Ward, J. Joyner, C. Lickfold, Y. Guo, M. Bennamoun, A practical tutorial on graph neural networks, ACM Computing Surveys (CSUR) 54 (2022) 1–35.
[19] X. Zhu, G. Tang, P. Wang, C. Li, J. Guo, S. Dietze, Dynamic global structure enhanced multi-channel graph neural network for session-based recommendation, Information Sciences 624 (2023) 324–343.
[20] H. Guo, R. Tang, Y. Ye, F. Liu, Y. Zhang, A novel knn approach for session-based recommendation, in: Advances in Knowledge Discovery and Data Mining: 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part II 23, Springer, 2019, pp. 381–393.
[21] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.
[22] S. Okura, Y. Tagami, S. Ono, A. Tajima, Embedding-based news recommendation for millions of users, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 1933–1942.