BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 74(3), 2026, Article number: e158304
DOI: 10.24425/bpasts.2026.158304

GraSS: A graph-based skip-gram synthesizer for session data augmentation

Viet Anh NGUYEN[1], Dang Son NGUYEN[1]*, Thi Nhan VU[1], Thi Lan Anh VU[1], and Thi Ngoc Tu NGUYEN[2]

[1] Institude of Information Technology – Vietnam Academy of Science and Technology, Viet Nam
[2] Electric Power University, Viet Nam

Abstract. Session-based recommender systems using graph neural networks (GNNs) have achieved strong performance by modeling item transitions within user sessions. Despite their success, these models often struggle to generalize well to infrequent or uncommon behavior patterns. Such patterns, due to their limited representation in the training data, are typically overshadowed by frequent transitions, leading to biased recommendations and reduced performance in real-world scenarios where long-tail behaviors are prevalent. To address this limitation, we introduce GraSS (graph-based skip-gram synthesizer), a novel data augmentation framework designed to improve the robustness of GNN-based session recommenders. GraSS identifies sessions containing rare item transitions and enriches them by generating synthetic session sequences. This is achieved by utilising skip-gram statistics to capture contextual item co-occurrences and applying random walks on an item graph to generate plausible but diverse session paths. The augmented sessions are then used to retrain the model, enabling better learning from sparse behaviors. Experiments on standard session-based recommendation benchmarks demonstrate that GraSS consistently improves recommendation accuracy.

Keywords: recommender system; graph neural network; data augmentation; user behavior modeling.

1. INTRODUCTION

Session-based recommender systems aim to predict the next item a user will interact with based solely on their current session, without relying on persistent user identities or long-term history. This formulation is particularly useful in privacy-sensitive environments and cold-start scenarios, such as e-commerce or media platforms where users frequently browse anonymously. Graph neural networks (GNNs) have become a leading approach in this domain. By modeling sessions as graphs where items are nodes and transitions are edges, GNN-based models can capture both local item relationships and the broader structure of user behavior. This allows them to outperform traditional sequential models, especially in capturing complex interaction patterns within sessions.

However, a critical limitation remains: GNNs tend to overfit to frequent item transitions. Since these models rely on node embeddings learned from co-occurrence patterns, rare or infrequent transitions, such as interactions with niche or new items, are often underrepresented in the learned representations. This leads to biased predictions that favor popular items, while failing to generalize to uncommon but meaningful behaviors. These rare transitions, although less frequent, often carry important signals. For example, they may indicate a user exploring unfamiliar categories, showing interest in emerging trends, or deviating from standard browsing patterns. Ignoring such patterns reduces the system ability to personalize recommendations effectively, especially in diverse or sparse data environments.

To address this challenge, we propose GraSS (graph-based skip-gram synthesizer), a data augmentation framework designed to improve the representation of rare session behaviors. GraSS identifies sessions dominated by infrequent item transitions using skip-gram co-occurrence statistics, constructs a graph from these sessions, and performs weighted random walks to generate synthetic sequences. These augmented sessions are incorporated into the training set, allowing the GNN model to better learn from underrepresented patterns without modifying its core architecture.

We validate GraSS on two widely used benchmark datasets, Diginetica and Yoochoose, and demonstrate that it consistently improves top-k recommendation performance. Our contributions include:

- Highlight and formalize the problem of frequency bias in GNN-based session recommendation.
- Introduce a skip-gram-based graph construction and sampling method to synthesize rare behavior sequences.
- Show that our augmentation approach improves the model performance without changing the underlying recommender architecture.

2. RELATED WORKS

2.1. Recommender systems

As e-commerce continues its rapid expansion, developing effective personalized product recommendations while managing computational resources became a critical research area, with

*e-mail: sonnd@ioit.ac.vn

Manuscript submitted 2025-08-11, revised 2026-01-29, initially accepted for publication 2026-03-13, published in May 2026.

a strong focus on tailoring suggestions to individual users. In 2004, Yoon Ho Cho introduced purchase recommendations by tracking customer shopping behavior on the web and classifying products using collaborative filtering (WebCF-PT) [1]. This work addressed data sparsity and scalability issues in e-commerce. Experiments showed WebCF-PT was about 18 times faster than conventional collaborative filtering. In 2007, Uwe Leimstoll proposed a collaborative filtering method based on similar customer behavior to provide personalized suggestions for online shops [2]. This method classified customers at the product category level and then analyzed customer carts within each category to predict suitable item groups, reducing the user-item matrix dimensionality and speeding up processing on large datasets without compromising accuracy. Heng-Tze Cheng *et al*. proposed a wide and deep learning solution for recommendations on Google Play [3]. This model combines the generalization capability of wide linear models with the feature interaction learning of deep neural networks, achieving better discovery of latent features. Experimental results showed significant improvements in app acquisitions compared to wide-only or deep-only models. This approach has been widely adopted by other e-commerce platforms.

To provide more relevant recommendations and address the limitations of traditional methods – such as data sparsity and scalability in collaborative filtering [4–6] and overspecialization or the cold-start problem in content-based filtering [7] – researchers have explored session-based approaches. Session-based recommender systems predict a user's next action by analyzing the sequence of items interacted with within the current session, without needing historical user data or content similarity analysis.

The increasing prominence of deep learning created new opportunities for innovation in recommendation research. The paper [8] applied recurrent neural networks (RNNs) to session-based recommendations for short sessions, specifically using gated recurrent units (GRU) to mitigate the vanishing gradient problem common in traditional RNNs. Their model handled large item sets effectively through sampling and parallel processing. In 2018, the paper [9] focused on improving performance by refining the loss function to better optimize the ranking of top recommended items, rather than the entire item list. Chatzis *et al*. [10] addressed the lack of user information in session-based systems by combining RNNs with a mechanism to learn latent variables within each session. These variables represent unobserved factors influencing user behavior changes across sessions, allowing the model to capture intra-session patterns without relying on user history. Recent work has also explored hybrid neural architectures for sequential recommendation. For example, Celik and Omurca propose SkipGT [11], a model that integrates skip-gram embeddings with a Transformer encoder to enhance next-item prediction in session-based recommendation systems

More recently, graph-based models have gained traction due to their ability to model complex item transitions. GC-San [12] combined graph neural networks (GNNs) with a self-attention mechanism. They represented sessions as graphs, learning local relationships (adjacent nodes) via GNNs and global relationships (across the session) via self-attention. These were linearly combined to predict the user's next action. SR-GNN [13] used GNNs to predict the next user action within a session. Each session was converted into a directed graph where edge weights reflected transition frequency. Session representations combined global preferences and current interests using an attention network to predict the next clicked item. However, a common issue with these GNN-based systems, due to their reliance on node embeddings, is difficulty recommending items involved in infrequent interactions.

Despite these advancements, a common thread across GNN-based approaches is their reliance on frequently observed transitions. Since these models learn embeddings driven by item co-occurrence, they tend to favor high-frequency interactions and underrepresent rare behaviors-an issue especially problematic for long-tail or exploratory user actions. This frequency bias limits the ability of models to generalize in diverse or sparse settings.

2.2. Session data augmentation

To combat data sparsity and enhance model generalization, researchers have proposed various augmentation strategies for session data. Many recent works focus on temporal consistency and structural realism when transforming sequences. Dang *et al*. [14] introduced TiCoSeRec, which uses timestamp-based cropping to preserve meaningful temporal patterns. This was extended by Dang *et al*. [15] through time interval-aware transformations, further enhancing sequence quality. RepPad [16] offers an alternative to zero-padding by using repeated elements from the original sequence, improving training dynamics for short sessions. Beyond simple transformations, some methods introduce more context-aware augmentations. Li *et al*. [17] proposed using basket-aware context to generate sequences that reflect item co-occurrence within shared transactional events. This approach enriches session data by modeling contextual affinity between items, rather than relying solely on sequential order. Zhou *et al*. [18] provided a systematic analysis of common augmentation techniques-cropping, masking, reordering, insertion, and deletion-and evaluated their effectiveness across various recommendation settings. While these methods improve robustness, they are largely applied uniformly across sessions and often overlook the structural and statistical nuances of rare interactions. Furthermore, applying such augmentations directly to graph-structured data introduces challenges in maintaining topological consistency.

All things considered, while recent advances in session-based modeling and data augmentation have improved performance and robustness, few works have directly addressed the frequency bias in GNN-based recommenders. Most augmentation methods operate at the sequence level without targeting sessions dominated by rare transitions. In contrast, our proposed method, GraSS, explicitly identifies infrequent transition patterns using skip-gram statistics and generates graph-consistent synthetic sequences through random walks. This targeted augmentation enables better representation of underrepresented session behaviors without modifying the underlying GNN architecture.

In this context, our proposed method, GraSS, introduces a

new perspective by approaching the augmentation problem from a graph-structural angle. Rather than modifying the model or relying on heavy generative machinery, GraSS intervenes directly at the level of session-graph construction. By identifying infrequent transition patterns using skip-gram statistics and selectively enriching their neighborhoods via controlled random walks, GraSS provides a lightweight mechanism for reducing item-frequency imbalance. This targeted augmentation enhances the representational strength of rare items without altering the underlying GNN architecture, offering a plug-and-play strategy that is both simple and effective.

For perspective, consider generative augmentation approaches such as diffusion-based sequence reconstruction models (e.g., DiffuASR [19]). These methods rely on probabilistic forward–reverse processes to reshape item transitions and typically involve additional parameters, complex training procedures, and careful hyperparameter tuning. Mentioning DiffuASR here serves only to illustrate contrast: whereas diffusion-based methods model uncertainty through a full generative framework, GraSS focuses on structural refinement through minimally invasive, graph-consistent perturbations. This distinction highlights GraSS core contribution-a principled yet lightweight augmentation method tailored specifically to the statistical weaknesses of session-graph data.

3. PROPOSED METHOD

3.1. Problem statement

This section establishes the fundamental definitions and problem formulation central to session-based recommendation and its graph representation.

Let $\mathbb{V} = \{v_1, v_2, \ldots, v_m\}$ be the set of all unique items present in the dataset.

A user session $\mathscr{S}$ is defined as a time-ordered sequence of interactions performed by a user within a specific timeframe. It captures a continuous stream of a user's behavior, such as viewing or clicking items. Formally, a session is represented as $\mathscr{S} = [v_1, v_2, \ldots, v_t]$ where each item $v_i \in \mathbb{V}$ was interacted with by a user, ordered chronologically from the first interaction (v_1) to the last (v_t) in that session. These sessions capture consecutive user actions, such as viewing products on an e-commerce site or watching videos on a streaming platform.

The objective of session-based recommendation is to predict the next item, $v_{t+1} \in \mathbb{V}$, that a user is most likely to interact with, given only their current session $\mathscr{S} = [v_1, v_2, \ldots, v_t]$. The outcome is typically a ranked list of candidate items from $\mathbb{V}$, ordered according to a model predicted probability or relevance score. The top-k items from this ranked list are then presented to the user as recommendations.

3.2. Skip-gram-based infrequent transition sessions identification and sequence generation via random walk

We propose **GraSS** (graph-based skip-gram synthesizer), a data augmentation framework designed to address the scarcity of infrequent transitions in session-based recommendation. As illustrated in Fig. 1, the overall workflow consists of three tightly coupled stages: (1) estimating global skip-gram statistics from observed sessions, (2) identifying sessions dominated by rare transitions, and (3) synthesizing new sequences via random walks over a skip-gram graph constructed specifically from these rare behaviors. Algorithm 1 provides a formal description of this procedure.

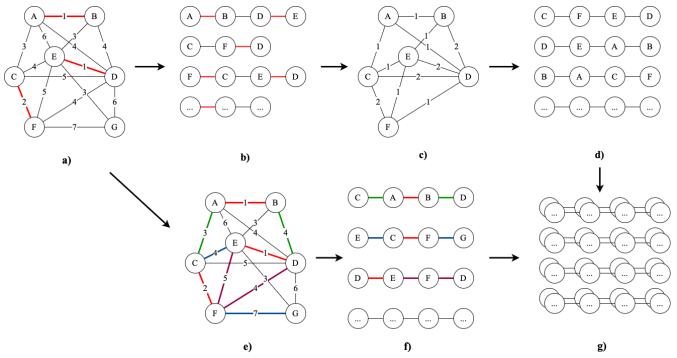

Fig. 1. Overall Workflow to Synthesize Data: (a) Global Skipgram Graph, Count Transition Frequency; (b) Identified Rare Behavior Sessions; (c) Build Isolated Skip-gram Graph; (d)Synthesize Data via Random Walk (GraSS); (e) Identified Low-frequency Item Transitions; (f) Synthesize Data via Random Walk Transition-Centric; (g) Final Synthetic Data

Algorithm 1. Synthesizing clickstream sequences via skip-gram graph sampling

Require: Clickstream sessions $\mathscr{S}$, window size w, walks per node k
Ensure: Synthesized sessions $\mathscr{S}_{\text{syn}}$

$L_{\text{avg}} \leftarrow$ average session length over $\mathscr{S}$
Initialize skip-gram count dictionary $\mathscr{C}$
for all $s \in \mathscr{S}$ **do**
  for $i = 1$ to $|s|$ **do**
    for $j = i+1$ to $\min(i+w, |s|)$ **do**
      $\mathscr{C}[(v_i, v_j)]$ += 1
    end for
  end for
end for
For each $s \in \mathscr{S}$, compute $\bar{c}_s \leftarrow$ average skip-gram count from $\mathscr{C}$
$\bar{c}_{\text{global}} \leftarrow$ mean of all $\bar{c}_s$ values
$\mathscr{S}_{\text{low}} \leftarrow \{s \in \mathscr{S} \mid \bar{c}_s < \bar{c}_{\text{global}}\}$
Build graph $G = (V, E)$ from skip-grams in $\mathscr{S}_{\text{low}}$ with edges weighted by $\mathscr{C}$
Initialize $\mathscr{S}_{\text{syn}} \leftarrow \emptyset$
for all $v \in V$ **do**
  for $t = 1$ to k **do**
    Generate random walk of length L_{avg} from v in G
    Add walk to $\mathscr{S}_{\text{syn}}$
  end for
end for
return $\mathscr{S}_{\text{syn}}$

We begin by processing the full set of clickstream sessions $\mathscr{S}$ to compute skip-gram co-occurrence statistics within a window of size w. As shown in Lines 3–9 of Algorithm 1, for each session $s \in \mathscr{S}$, every ordered item pair (v_i, v_j) that appears within distance w is counted and accumulated in a global dictionary $\mathscr{C}$. These counts provide a dataset-level estimate of short-range item transition frequency and serve as the statistical foundation for identifying infrequent behaviors.

Next, GraSS identifies sessions characterized by rare transitions. For each session s, we compute its average skip-gram frequency $\bar{c}_s$ by averaging the counts of all skip-gram pairs appearing in that session using the statistics stored in $\mathscr{C}$ (Line 10). We then compute the global mean skip-gram frequency $\bar{c}_{\text{global}}$ across all sessions (Line 11). Sessions whose average skip-gram frequency falls below this global mean are classified as low-occurrence sessions and collected into the set $\mathscr{S}_{\text{low}}$ (Line 12). This criterion allows GraSS to isolate sessions composed primarily of infrequent or weakly observed transitions rather than relying solely on raw item frequency.

To synthesize new data, GraSS constructs a dedicated skip-gram graph $G = (V, E)$ using only the transitions observed in $\mathscr{S}_{\text{low}}$ (Line 13). Nodes represent items appearing in low-occurrence sessions, and edges represent skip-gram co-occurrences weighted by their corresponding counts in $\mathscr{C}$. Although session data are inherently sequential, we intentionally construct the skip-gram graph with undirected edges to better capture the flexible and exploratory nature of user interest transitions during data augmentation. In real browsing behavior, particularly in sessions dominated by infrequent interactions, item transitions do not always reflect a strict or irreversible progression of intent. Instead, a transition between two items often represents a latent semantic or functional association rather than a fixed temporal order. For example, while one user may navigate from a kitchen appliance to a home decor item, another user with a similar intent may explore these items in the reverse order depending on individual browsing strategies.

Using directed edges at this stage would force an artificial asymmetry that is largely driven by sparsity rather than true user preference, especially for item pairs that appear only a few times. By treating edges as undirected, the random walk is allowed to move more freely between related items and to reflect how users may explore similar products in different orders. This helps the synthesizer focus on capturing general interest relationships instead of overfitting to noisy direction patterns. When training the downstream recommendation model, directionality is fully preserved through the construction of directed session graphs, as described in Section 3.4.

Finally, synthetic sequences are generated via weighted random walks on the skip-gram graph G. As specified in Lines 15–20 of Algorithm 1, multiple walks are initiated from each node, with transition probabilities proportional to skip-gram edge weights. Each walk has a length equal to the average session length L_{avg} computed from the original dataset (Line 1), ensuring that synthesized sessions are comparable in scale to real ones. The resulting set of synthetic sequences $\mathscr{S}_{\text{syn}}$ captures diverse yet representative transition patterns drawn from sparse regions of the interaction space.

The synthesized sequences are then combined with the original session set $\mathscr{S}$ to form an augmented training corpus. This enriched dataset improves coverage of underrepresented behaviors and is subsequently used to construct the session graphs for training the GNN-based recommender model.

3.3. Transition-centric skip-gram synthesis

In addition to identifying sessions dominated by low-frequency transitions, we introduce a complementary augmentation strategy that operates at the transition level. While GraSS focuses on global session rarity, this method targets individual low-frequency transitions across the entire dataset, regardless of the session in which they appear. These transitions, though embedded within frequent sessions, are often underrepresented in the learning process due to their sparse occurrence (Algorithm 2).

Algorithm 2. Synthesizing sequences centered on low-frequency transitions

Require: Global skip-gram graph $G = (V, E)$ with edge weights $\mathscr{C}$, threshold τ, walk length L_w, walks per transition k
Ensure: Transition-based synthetic sequences $\mathscr{S}_2$
1: Initialize $\mathscr{T}_{\text{low}} \leftarrow \{(v_i, v_j) \in E \mid \mathscr{C}[(v_i, v_j)] < \tau\}$
2: Initialize $\mathscr{S}_2 \leftarrow \emptyset$
3: **for all** $(v_i, v_j) \in \mathscr{T}_{\text{low}}$ **do**
4: **for** $t = 1$ to k **do**
5: Perform weighted random walk of length L_w **backward** from v_i to get LeftSeq
6: Perform weighted random walk of length L_w **forward** from v_j to get RightSeq
7: Concatenate sequence: $s_{\text{syn}} \leftarrow \text{LeftSeq} \parallel [v_i, v_j] \parallel \text{RightSeq}$
8: Add s_{syn} to $\mathscr{S}_2$
9: **end for**
10: **end for**
11: **return** $\mathscr{S}_2$

We begin by constructing the same global skip-gram co-occurrence graph $G = (V, E)$ as described earlier, where each edge $(v_i, v_j) \in E$ is assigned a weight c_{ij} based on its co-occurrence frequency within the skip-gram window. We define a tunable threshold parameter τ to identify low-frequency transitions. Any transition (v_i, v_j) such that $c_{ij} < \tau$ is considered infrequent and selected for augmentation. In our experiments, we set τ to be the global average edge weight $\bar{c}_{\text{global}}$. Let the set of all such transitions be $\mathscr{T}_{\text{low}}$.

For each low-frequency transition $(v_i, v_j) \in \mathscr{T}_{\text{low}}$, we generate synthetic clickstream sequences that embed this transition within broader context. Specifically, we initiate a random walk from node v_i to the left and a random walk from node v_j to the right, each of length L_w, where L_w is a hyperparameter approximating half the average session length. These walks are performed on the global graph G, with transition probabilities proportional to edge weights. The left walk yields a sequence $[u_{L_w}, \ldots, u_1]$ terminating at v_i, while the right walk yields a sequence $[w_1, \ldots, w_{L_w}]$ starting from v_j. The resulting synthetic

sequence is formed by concatenating the reversed left walk, the fixed transition (v_i, v_j), and the right walk:

$$s_{\text{syn}} = \left[u_{L_w}, \ldots, u_1, v_i, v_j, w_1, \ldots, w_{L_w}\right].$$

Multiple such sequences can be generated per transition by performing multiple random walks with different sampling seeds. These sequences are then added to the synthetic set $\mathscr{S}_2$, designed to explicitly reinforce the model exposure to underrepresented transitions.

Finally, we merge this new set of synthetic sequences with those generated by GraSS to form a comprehensive augmented dataset. The final training set used to construct session graphs and train the GNN model becomes:

$$\mathscr{S}_{\text{aug}} = \mathscr{S} \cup \mathscr{S}_1 \cup \mathscr{S}_2,$$

where $\mathscr{S}_1$ is the augmented set from session-level GraSS synthesis, and $\mathscr{S}_2$ is the new set generated from rare transition expansion. This combination ensures that both globally rare session patterns and locally infrequent transitions are adequately captured during model training, providing improved robustness and generalization in the presence of long-tail behaviors.

3.4. Session graphs construction

To represent the flow and relationships among items within a session more comprehensively than a simple linear list, each user session $\mathscr{S} = [v_1, v_2, \ldots, v_t]$ is represented as a directed graph $G_{\mathscr{S}} = (V_{\mathscr{S}}, E_{\mathscr{S}})$. In this graph:

- $V_{\mathscr{S}}$: The set of unique items appearing in the session $\mathscr{S}$. Thus, $V_{\mathscr{S}} \subseteq \mathbb{V}$.
- $E_{\mathscr{S}}$: The set of directed connections showing sequential transitions between items within $\mathscr{S}$. A directed edge $(u, v) \in E_{\mathscr{S}}$ exists if item v was interacted with immediately after item u in the sequence $\mathscr{S}$.

Connections in $E_{\mathscr{S}}$ can be given weights to show how often specific transitions occur within the session. A standard way to set the weight of an edge (v_i, v_j) is by counting how many times the transition from v_i to v_j happens consecutively within session $\mathscr{S}$.

3.5. Item embedding

After constructing the session graphs, each node $v_1 \in \mathbb{V}$ is mapped into a shared embedding space. This results in a vector representation $\mathbf{v}_i \in \mathbb{R}^d$, where d denotes the embedding dimension. These node embeddings serve as the basis for representing entire sessions. Graph neural networks (GNNs) are then employed to generate these node representations by utilizing the structural information inherent in graph topologies, which allows them to capture intricate relationships among items. This makes GNNs especially well-suited for session-based recommendation tasks. In our approach, we specifically utilize gated recurrent units, a GNN variant proposed in 2015, to learn the item embeddings. The update rule for a node $v_{s,i}$ in the session graph G is formally defined as follow:

$$a_{s,i}^{(t)} = A_{s,i:}[\mathbf{v}^{t-1}, \ldots, \mathbf{v}_{s_n}^{t-1}]^\top H + b, \tag{1}$$

$$z_{s,i}^{(t)} = \delta(W_z a_{s,i}^{(t)} + U_z \mathbf{v}^{(t-1)}), \tag{2}$$

$$r_{(s,i)}^{(t)} = \delta(W_r a_{s,i}^{(t)} + U_r \mathbf{v}_i^{(t-1)}), \tag{3}$$

$$\widetilde{\mathbf{v}_i(t)} = \tanh(W_O a_{s,i}^{(t)} + U_O(r_{s,i}^{(t)} \odot \mathbf{v}^{(t-1)})), \tag{4}$$

$$\mathbf{v}_i^{(t)} = (1 - z_{s,i}^{(t)}) \odot \mathbf{v}^{(t-1)} + z_{s,i}^{(t)} \odot \widetilde{\mathbf{v}_i^{(t)}}. \tag{5}$$

Here, t denotes the current training step. The matrix $A_{s,i} \in \mathbb{R}^{1\times 2n}$ represents the i-th row of the adjacency matrix A_s corresponding to node $v_{s,i}$. The parameters $H \in \mathbb{R}^{d\times 2d}$ and $b \in \mathbb{R}^d$ denote the weight matrix and bias vector, respectively. The vectors $v_1^{(t-1)}, \ldots, v_{s_n}^{(t-1)}$ correspond to the node representations from the previous time ste within the session graph. The matrices $z_{s,i} \in \mathbb{R}^{d\times d}$represent the update and reset gates, respectively. The function $\delta(.)$ denotes the sigmoid function, and $\odot$ indicates element-wise multiplication. In each session graph G, the gated graph neural network propagates information among neighboring nodes. The reset and update gates determine which parts of the incoming information should be retained or discarded during the update process.

3.6. Session embedding

The session graph is then embedded using a graph neural network (GNN) model. The resulting session representation comprises two main components:

Local Embedding (s_l): Captures the user's most recent interest. This is often represented by the embedding of the last item in the session sequence, $\mathbf{v}_n$:

$$s_l = \mathbf{v}_n.$$

Global embedding (s_g): Represents the user's overall interest throughout the session. This is usually derived by aggregating the embeddings of all items in the session, often weighted by an attention mechanism:

$$s_g = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i,$$

where the attention weight α_i for item v_i is calculated as:

$$\alpha_i = q^T f(W_1 \mathbf{v}_{s,n} + W_2 \mathbf{v}_i + c).$$

Here,

- α_i: the attention weight for item v_i.
- $f(W_1 \mathbf{v}_{s,n} + W_2 \mathbf{v}_i + c)$: an activateion function (e.g., sigmoid or tanh), and W_1, W_2, q, c are learnable parameters of the attention network.

The final session representation s_h is obtained by combining the local and global embeddings, ensuring both recent and overall session context are captured:

$$s_h = W_3[s_l; s_g],$$

where W_3 is a learnable linear transformation matrix, and $[s_l; s_g]$ denotes the concatenation of the local and global embedding vectors.

3.7. Behavior prediction

Once the session representation s_h is obtained, the next step is to predict the user's next likely interaction.

The prediction score z_i for each candidate item v_i (from the entire item set $\mathbb{V}$) is computed, typically using the dot product between the session representation s_h and the candidate item's embedding $\mathbf{v}_i$:

$$\hat{z}_i = s_h^T \mathbf{v}_i \,.$$

A softmax function is applied over the scores z for all candidate items to convert them into a probability distribution:

$$\hat{y} = softmax(\hat{z}),$$

where $\hat{y}$ represents the probability distribution over all candidate items, where y_i indicates the predicted likelihood that item v_i will be the next item clicked/interacted with in the session.

4. EXPERIMENTAL RESULTS

4.1. Datasets and evaluation metrics

To evaluate the effectiveness of our proposed GraSS method, we conducted comprehensive experiments on two widely recognized benchmark datasets commonly used in session-based recommendation research: Diginetica and Yoochoose.

The **Diginetica dataset** originates from the CIKM Cup 2016 competition and comprises real-world interaction data from an e-commerce platform. It serves as a standard resource for evaluating session-based models, particularly those focusing on sequential user behavior. The dataset contains information on 184 047 unique items, 232 816 users, 1 235 380 views, and 18 025 purchases.

The **Yoochoose dataset**, derived from the RecSys Challenge 2015, consists of a large volume of user clickstream data collected over six months from an e-commerce website. Its substantial size and focus on click sequences make it a frequent choice for benchmarking session-based recommendation algorithms. This dataset includes 9 249 729 sessions, 33 004 944 clicks, and 52 739 unique items. Due to the substantial scale of the Yoochoose dataset, we follow a widely adopted experimental protocol in the session-based recommendation literature and conduct our experiments on the Yoochoose 1/64 subset. This subset is constructed by selecting sessions from a restricted time period based on their timestamps, resulting in approximately one sixty-fourth of the full dataset. Using the Yoochoose 1/64 split significantly reduces computational cost while preserving representative user behavior patterns, and it enables fair and direct comparison with numerous prior studies that report results on the same subset.

Prior to training, both datasets underwent standard preprocessing steps to ensure data quality and focus on meaningful interactions. We filtered out sessions containing fewer than 3 items to remove uninformative short sequences. Additionally, items appearing less than 5 times across the entire dataset were removed. Following common practice in session-based learning [20], we transformed the filtered sessions into sequence-label pairs suitable for supervised model training. For each session $\mathscr{S} = [v_1, v_2, \ldots, v_t]$, we generated input-output pairs of the form $([v_1, \ldots, v_i], v_{i+1})$ for all valid prefixes i from 1 to $t-1$. These pairs constitute the training and testing instances used for predicting the next item in a sequence. After preprocessing, the statistics of the datasets used in our experiments are summarized in Table 1.

Table 1
Statistics of the datasets used in experiments

	Yoochoose1/64	Diginetica
#session	425 757	780 328
avg session length	5.12	6.16
#train session	369 859	719 470
#test session	60 585	55 898

We evaluate the effectiveness of the proposed method using two standard metrics: Recall and MRR (mean reciprocal rank). Higher values for these metrics indicate better model performance in recommending relevant items and ranking them highly in the suggestion list.

Recall measures the proportion of relevant items that are successfully recommended by the system out of the total number of actual relevant items. In recommender systems, Recall is typically calculated at a cutoff k (e.g., Recall@20), meaning it considers only the top-k recommended items.

$$Recall@k = \frac{1}{N} \sum_{s \in \mathscr{S}} \frac{|\text{Top-k recommended items}|}{|\text{Item user interacted with}|} \,.$$

MRR (mean reciprocal rank) evaluates the ranking quality by considering the position (rank) of the first correctly recommended relevant item in the ranked list.

$$MRR@k = \frac{1}{|\mathscr{S}|} \sum_{s \in \mathscr{S}} \frac{1}{rank_s} \,,$$

where

- $|\mathscr{S}|$: the total number of sessions in the test set.
- $rank_s$: the rank position of the first relevant item in the top-k recommendation list for session s. If no relevant item is found within the top-k, the reciprocal rank for that session is 0.

To assess the performance of our proposed algorithm, we compare it against the following algorithms:

- Item-KNN [21]: Recommends items similar to the previously clicked item in the session. Similarity is typically defined using cosine similarity between item vectors derived from session co-occurrences.
- FPMC [22]: A sequential prediction method based on factorized Markov chains.
- GRU4REC [8]: Uses recurrent neural networks (specifically GRUs) to model user session sequences for next-item prediction.

- NARM [6]: Employs an RNN architecture with an attention mechanism to capture sequential information and the user's main purpose within the session.
- STAMP [23]: Models short-term interests based on the last click, along with general interests derived from the session context.
- SR-GNN [13]: A session-based recommendation model utilizing graph neural networks.
- TA-GNN [24]: A target attentive graph neural network (TAGNN) model for session-based recommendation.

To evaluate the effectiveness of our proposed model, we compare its performance against existing session-based recommendation methods using Recall and MRR metrics, measured at $k = 10$ and $k = 20$. The best results are highlighted in bold (Table 2).

Table 2
Comparison of the proposed method against baseline/existing methods

Diginetica				
Method	Recall@10	MRR@10	Recall@20	MRR@20
Item-KNN	25.07	10.77	35.75	11.57
FPMC	15.43	6.20	26.53	6.95
GRU4REC	17.93	7.33	29.45	8.33
NARM	35.44	15.13	49.70	16.17
STAMP	33.98	14.26	45.64	14.32
SR-GNN	36.86	15.52	50.73	17.59
TA-GNN	–	–	51.31	18.03
GraSS+SR-GNN	38.93	17.11	51.66	17.99
GraSS+TA-GNN	**39.67**	**17.53**	**52.63**	**18.42**

Yoochoose1/64				
Method	Recall@10	MRR@10	Recall@20	MRR@20
Item-KNN	41.82	21.24	51.60	21.81
FPMC	35.38	14.57	45.62	15.01
GRU4REC	50.30	22.81	60.64	22.89
NARM	57.56	27.16	68.32	28.63
STAMP	58.67	29.66	68.74	29.67
SR-GNN	60.04	30.08	70.57	30.94
TA-GNN	–	–	71.02	**31.12**
GraSS+SR-GNN	60.66	**30.24**	70.99	30.97
GraSS+TA-GNN	**60.82**	29.87	**71.55**	30.63

Among the traditional methods, experimental results indicate that Item-KNN, which recommends items based on their similarity to previously interacted, with items in the session, outperforms the Markov chain, based FPMC. However, Item-KNN limitation lies in considering only item-item similarity, potentially overlooking broader user interests captured within the session sequence.

With the adoption of deep learning approaches, methods like GRU4REC, NARM, STAMP, and SR-GNN achieve superior results compared to the traditional baselines. GRU4REC was the first deep learning model applied to session-based recommendation, utilizing recurrent neural networks (specifically the GRU architecture) to process sequential data. NARM and STAMP incorporate self-attention mechanisms, often focusing more on recent user interactions. Experimental results show that NARM and STAMP significantly outperform GRU4REC. SR-GNN, which models sessions using graph structures, achieves further performance improvements. This demonstrates that employing graph neural networks for session-based recommendation is a promising research direction.

With its focus on infrequent transition sequences, GraSS concentrates on handling infrequent interactions to improve the model's recommendation quality. We utilize Skip-gram combined with random walk to generate augmented data for sequences identified as containing infrequent transition. Experimental results demonstrate the effectiveness of our proposed method on both the Diginetica and Yoochoose 1/64 datasets.

Tables 3 and 4 present results showing the impact of varying the number of random walk steps for the Yoochoose 1/64 and Diginetica datasets, respectively. Optimal performance on Diginetica is achieved with random_walk = 2, while Yoochoose reaches its peak performance with random_walk = 7. This suggests that dataset characteristics, such as average sequence length and the prevalence of rare interactions, significantly influence the determination of the optimal number of random walk steps.

Table 3
Analysis of the effect of random walk steps on the Yoochoose 1/64 dataset with SR-GNN baseline

	3	4	5	6	7	8
Recall@10	60.40	60.62	60.52	60.50	**60.66**	60.58
MRR@10	30.13	30.09	30.06	30.19	**30.24**	30.23
Recall@20	70.81	70.99	70.93	71.03	70.99	**71.05**
MRR@20	30.86	30.83	30.79	30.92	**30.97**	30.97

Table 4
Analysis of the effect of random walk steps on the Diginetica dataset with SR-GNN baseline

	1	2	3	4	5	6	7
Recall@10	38.86	**38.93**	38.74	38.71	38.66	38.47	38.28
MRR@10	17.03	**17.11**	17.01	16.92	16.94	16.81	16.74
Recall@20	51.47	**51.66**	51.68	51.66	51.44	51.43	51.15
MRR@20	17.90	**17.99**	17.91	17.81	17.82	17.71	17.63

GraSS achieves strong performance across both Recall and MRR metrics. Particularly on the Diginetica dataset, GraSS yields high performance for top-10 recommendations and provides substantial improvements for top-20 recommendations. Conversely, on the Yoochoose 1/64 dataset, characterized by

longer average session lengths, the performance gain over SR-GNN is less pronounced. This suggests that GraSS is particularly well-suited for datasets characterized by shorter sessions. Investigating improvements for datasets with longer sequences remains an area for future work.

4.2. Effect of GraSS, complementary augmentation, and their combination

To further understand the impact of our proposed augmentation strategies, we evaluate three variants – *GraSS* alone, a *complementary augmentation* strategy (Comp.Aug), and their *combined* use – under two backbone models (SR-GNN and TA-GNN) and two datasets (Yoochoose 1/64 and Diginetica). The evaluation focuses on Recall@K and MRR@K, which together reflect coverage and ranking quality of the predicted next items.

Results on TA-GNN. On Yoochoose 1/64 (Table 5), GraSS consistently outperforms Comp.Aug across all metrics, indicating that GraSS more effectively enhances the model's ability to recover relevant items in short-session environments. The combined method (GraSS+Comp.Aug) achieves the highest MRR@10, suggesting improved ranking quality, although recall remains slightly lower than GraSS alone. On Diginetica, GraSS again provides the strongest improvements on Recall@10 and MRR@10; while the combined method slightly improves Recall@20, it does not surpass GraSS in ranking metrics. These findings suggest that combining both augmentations does not guarantee additional gains for TA-GNN, especially on datasets with short and heterogeneous sessions.

Table 5

Comparison of augmentation strategies (GraSS, complementary augmentation, and their combination) across SR-GNN and TA-GNN

Yoochoose 1/64				
Method	Recall@10	MRR@10	Recall@20	MRR@20
SR-GNN Baseline	60.04	30.08	70.57	30.94
SR-GNN + GraSS	**60.66**	**30.24**	**70.99**	**30.97**
SR-GNN + Comp. Aug	59.60	29.59	70.16	30.34
SR-GNN + GraSS + Comp. Aug	60.13	29.89	70.53	30.62
TA-GNN Baseline	–	–	71.02	31.12
TA-GNN + GraSS	**60.82**	29.87	**71.55**	30.63
TA-GNN + Comp. Aug	60.22	29.88	70.76	30.61
TA-GNN + GraSS + Comp. Aug	60.69	**30.01**	71.11	**30.74**

Results on SR-GNN. Across both datasets, all augmentation strategies improve the SR-GNN baseline. GraSS provides the most substantial improvements (as seen in Table 6), particularly on Diginetica, where it boosts Recall@10 by +5.62% and MRR@10 by +10.24%. Comp.Aug and the combined variant also yield improvements, but their gains are smaller and less consistent compared to GraSS. On Yoochoose 1/64, GraSS once again shows the most stable improvements across metrics.

Table 6

Comparison of augmentation strategies (GraSS, complementary augmentation, and their combination) on diginetica

Diginetica				
Method	Recall@10	MRR@10	Recall@20	MRR@20
SR-GNN Baseline	36.86	15.52	50.73	17.59
SR-GNN + GraSS	**38.93**	**17.11**	**51.66**	**17.99**
SR-GNN + Comp. Aug	37.95	16.76	50.50	17.63
SR-GNN + GraSS + Comp. Aug	38.08	16.86	50.86	17.74
TA-GNN Baseline	–	–	51.31	18.03
TA-GNN + GraSS	**39.67**	**17.53**	**52.63**	**18.42**
TA-GNN + Comp. Aug	39.06	17.41	51.54	18.27
TA-GNN + GraSS + Comp. Aug	39.19	17.41	51.82	18.27

Summary. These experiments demonstrate that targeted augmentation of rare transitions meaningfully enhances session-based recommendation. Among all strategies, GraSS consistently delivers the largest and most stable improvements, particularly on short and sparse datasets. Although combining augmentations can sometimes offer additional benefits, GraSS alone already captures most of the achievable gains. This highlights the effectiveness of focusing augmentation specifically on infrequent transitions rather than applying uniform transformations across sessions.

4.3. Statistical significance testing

To objectively assess whether incorporating GraSS leads to a statistically significant improvement over the baseline models (SR-GNN and TA-GNN), we conducted paired statistical tests on the performance metrics reported in Table 2. Since the GraSS variants are constructed directly on top of the baseline methods and evaluated on the same dataset splits, a paired setting is appropriate. We formally define the hypothesis as follows:

$$H_0 : \mathbb{E}\big[\,\text{GraSS} - \text{Baseline}\,\big] \leq 0,$$
$$H_1 : \mathbb{E}\big[\,\text{GraSS} - \text{Baseline}\,\big] > 0,$$

where the difference is computed for each matched metric value (MRR@20 and Recall@20/Precision@20), excluding Recall@10 and MRR@10 for TA-GNN due to the unavailability of these results in the original paper.

Paired *t*-test. A one-sided paired *t*-test was conducted to evaluate whether the mean performance difference is greater than zero. The test yields a test statistic of

$$t = 3.2398,$$

with a corresponding one-sided p-value of

$$p = 0.0039.$$

This result indicates statistical significance at the 5% level, supporting the hypothesis that GraSS improves model performance.

Wilcoxon signed-rank test. Because the number of paired observations is relatively small, we additionally performed a non-parametric Wilcoxon signed-rank test. The test statistic was

$$W = 72.0,$$

with a one-sided p-value of

$$p = 0.0034.$$

This result further corroborates the conclusion that GraSS provides a consistent and statistically significant improvement over the baseline methods.

Conclusion. Both the parametric and non-parametric tests show strong statistical evidence that adding GraSS leads to significant performance gains compared with the baseline models across the evaluated metrics.

4.4. Effect of GraSS augmentation on the Yoochoose1/64 dataset

To better understand how GraSS influences model behavior, we conducted an error analysis focusing on Yoochoose1/64 Dataset cases where the baseline model (trained on the original dataset) failed to predict the next item correctly, but the model trained on GraSS-augmented data succeeded.

As can be seen in Table 7, cross the test set, we identified a total of 1 469 such corrected cases. Remarkably, 1 404 of them – over **95%** – originated from low-occurrence sessions, i.e., sequences dominated by infrequent items or rare transition patterns. Only a small subset of corrected cases came from regular, high-frequency interaction patterns. These statistics suggest that GraSS primarily strengthens the model ability to handle underrepresented session behaviors, which aligns with its design objective of enriching the latent space surrounding rare transitions. While GraSS does not guarantee improvement for every individual rare session, the empirical evidence indicates that training with GraSS makes the model substantially more capable of resolving prediction errors rooted in data sparsity.

Table 7
Statistics of corrected prediction errors after applying GraSS on Yoochoose1/64

Statistic	Value
Total corrected cases	1 469
Corrected low-occurrence sessions	1 404
Corrected normal-frequency sessions	65
Percentage from low-occurrence sessions	95.6%

5. DISCUSSION

The computational overhead of GraSS arises primarily from the additional preprocessing and data synthesis steps introduced before model training. Specifically, computing skip-gram co-occurrence statistics over all sessions, constructing the skip-gram graph for low-occurrence transitions, and performing multiple weighted random walks to generate synthetic sequences add extra time and memory costs compared to training on the original data alone. However, these operations are performed offline and only once prior to model training, and their cost scales linearly with the number of sessions and skip-gram pairs. In practice, this overhead is modest relative to the overall training cost of GNN-based session recommenders and can be controlled through hyperparameters such as the skip-gram window size, the number of random walks per node, and the walk length.

The threshold τ used to identify low-occurrence sessions is a heuristic design choice and is set to the global average skip-gram weight computed over the training data. We understand that the choice for this hyperparameter is not theoretically optimal, and we recognize that alternative settings may influence the behavior and effectiveness of the data augmentation process. In this work, we adopt this setting as a simple and interpretable, data-driven starting point that distinguishes relatively infrequent transition patterns from common ones without introducing additional tunable hyperparameters. While this choice adapts naturally to datasets with different scales and sparsity levels, a more systematic investigation of this hyperparameter, including adaptive or learned alternatives, remains an important direction for future work.

Although GraSS yields smaller improvements on YooChoose 1/64, isolating the exact cause remains challenging. This behavior may be attributed to multiple interacting dataset properties, and a deeper analysis is required to draw definitive conclusions.

6. CONCLUSIONS

This paper addressed a key challenge in session-based recommendation using graph neural networks: their inherent difficulty in effectively learning from and recommending items involved in infrequent session interactions. Conventional GNN models often underrepresent or neglect the unique patterns within infrequent transition sessions, leading to biased recommendations towards popular items. We proposed GraSS (graph-based skip-gram synthesizer), a novel method designed to mitigate this limitation through targeted data augmentation. By analyzing item co-occurrence within sessions using a skip-gram approach and constructing a graph specifically from identified rare transitions, GraSS synthesizes additional training examples that provide the GNN model with a more balanced view of infrequent behaviors. Our approach yields a significant advantage: the resulting system not only maintains strong performance on commonly interacted-with items but also substantially improves its ability to accurately recommend items considered rare or those appearing in less frequent sequential patterns.

Experimental evaluation conducted on widely-used benchmark datasets demonstrated the effectiveness of GraSS. Our method consistently achieved superior recommendation performance for next-item prediction tasks (e.g., top-k recommen-

dation) compared to baseline GNN approaches that do not incorporate such specific data augmentation for rare interactions. Future work will focus on exploring alternative strategies for synthesizing infrequent transition sequences, investigating the application of GraSS with different graph neural network architectures, and evaluating its performance and adaptability across a wider variety of datasets exhibiting diverse types of sparsity and user behaviors. Continued optimization of the augmentation process remains a key area to further push the boundaries of prediction accuracy for both common and infrequent items.

ACKNOWLEDGEMENTS

This work was supported by the project "Research on the Application of Graph Theory in Customer Behavior Analysis", code CSCL02.01/24-25, Institute of Information Technology, Vietnam Academy of Science and Technology.

REFERENCES

[1] Y.H. Cho and J.K. Kim, "Application of Web usage mining and product taxonomy to collaborative recommendations in e-commerce," *Expert Syst. Appl.*, vol. 26, no. 2, pp. 233–246, 2004, doi: 10.1016/S0957-4174(03)00138-6.

[2] U. Leimstoll and H. Stormer, "Collaborative recommender systems for online shops," 2007.

[3] H.T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, and H. Shah, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop on Deep Learning for Recommender Systems*, 2016, pp. 7–10, doi: 10.1145/2988450.2988454.

[4] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective personalization based on association rule discovery from web usage data," in *Proc. 3rd International Workshop on Web Information and Data Management*, 2001, pp. 9–15, doi: 10.1145/502932.502935.

[5] Z. Zhang and O. Nasraoui, "Efficient hybrid Web recommendations based on Markov clickstream models and implicit search," in *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*. IEEE, 2007, pp. 621–627, doi: 10.1109/WI.2007.111.

[6] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proc. 2017 ACM Conference on Information and Knowledge Management*, November 2017, pp. 1419–1428, doi: 10.48550/arXiv.1711.04725.

[7] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, 2005, doi: 10.1109/TKDE.2005.99.

[8] B. Hidasi, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015, doi: 10.48550/arXiv.1511.06939.

[9] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proc. 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 843–852, doi: 10.1145/3269206.3271761.

[10] S.P. Chatzis, P. Christodoulou, and A.S. Andreou, "Recurrent latent variable networks for session-based recommendation," in *Proc. 2nd Workshop on Deep Learning for Recommender Systems*, 2017, pp. 38–45, doi: 10.1145/3125486.3125493.

[11] E. Celik and I. Omurca, "Skip-Gram and Transformer Model for Session-Based Recommendation," *Appl. Sci.*, vol. 14, no. 14, p. 6353, 2024, doi: 10.3390/app14146353.

[12] C. Xu, P. Zhao, Y. Liu, V.S. Sheng, J. Xu, F. Zhuang, and X. Zhou, "Graph contextualized self-attention network for session-based recommendation," in *IJCAI*, vol. 19, 2019, pp. 3940–3946, doi: 10.24963/ijcai.2019/547.

[13] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, 2019, pp. 346–353, doi: 10.1609/aaai.v33i01.3301346.

[14] Y. Dang, E. Yang, G. Guo, L. Jiang, X. Wang, X. Xu, and H. Liu, "TiCoSeRec: Augmenting data to uniform sequences by time intervals for effective recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 6, pp. 2686–2700, 2023, doi: 10.1109/TKDE.2023.3324312.

[15] Y. Dang, E. Yang, G. Guo, L. Jiang, X. Wang, X. Xu, and H. Liu, "Uniform sequence better: Time interval aware data augmentation for sequential recommendation," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4225–4232, doi: 10.1609/aaai.v37i4.25540.

[16] Y. Dang, Y. Liu, E. Yang, G. Guo, L. Jiang, X. Wang, and J. Zhao, "Repeated padding as data augmentation for sequential recommendation," *arXiv e-prints*, 2024, doi: 10.1145/3640457.3688110.

[17] M. Li, M. Ariannezhad, A. Yates, and M. De Rijke, "Masked and swapped sequence modeling for next novel basket recommendation in grocery shopping," in *Proc. 17th ACM Conference on Recommender Systems*, September 2023, pp. 35–46, doi: 10.48550/arXiv.2308.01308.

[18] P. Zhou, Y.L. Huang, Y. Xie, J. Gao, S. Wang, J.B. Kim, and S. Kim, "Is contrastive learning necessary? A study of data augmentation vs contrastive learning in sequential recommendation," in *Proc. ACM Web Conference 2024*, May 2024, pp. 3854–3863, doi: 10.1145/3589334.3645661.

[19] X.Z. Qidong Liu and R. Tang, "Diffusion Augmentation for Sequential Recommendation," in *32nd ACM International Conference on Information and Knowledge Management*, 2003, pp. 17–24, doi: 10.3115/1118935.1118938.

[20] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *Proc. 1st Workshop on Deep Learning for Recommender Systems*, 2016, pp. 17–22, doi: 10.1145/2988450.2988452.

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th International Conference on World Wide Web*, 2001, pp. 285–295, doi: 10.1145/371920.372071.

[22] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th International Conference on World Wide Web*, April 2010, pp. 811–820, doi: 10.1145/1772690.1772773.

[23] 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 International Conference on Knowledge Discovery & Data Mining*, July 2018, pp. 1831–1839, doi: 10.1145/3219819.3219950.

[24] F. Yu, Y. Zhu, Q. Liu, S. Wu, L. Wang, and T. Tan, "TAGNN: Target Attentive Graph Neural Networks for Session-based Recommendation," in *Proc. 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1921–1924, doi: 10.1145/3397271.3401319.