Global Context Enhanced Graph Neural Networks for Session-based Recommendation

Ziyang Wang[1], Wei Wei[1,⊠], Gao Cong[2], Xiao-Li Li[3], Xian-Ling Mao[4], Minghui Qiu[5]
[1] Cognitive Computing and Intelligent Information Processing (CCIIP) Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology
[2] School of Computer Engineering, Nanyang Technological University, Singapore
[3] Institute for Infocomm Research, Singapore
[4] School of Computer Science and Technology, Beijing Institute of Technology
[5] Alibaba Group
[1] {ziyang1997, weiw}@hust.edu.cn [2] gaocong@ntu.edu.sg [3] xlli@i2r.a-star.edu.sg
[4] maoxl@bit.edu.cn [5] minghuiqiu@gmail.com

ABSTRACT

Session-based recommendation (SBR) is a challenging task, which aims at recommending items based on anonymous behavior sequences. Almost all the existing solutions for SBR model user preference only based on the current session without exploiting the other sessions, which may contain both relevant and irrelevant item-transitions to the current session. This paper proposes a novel approach, called **G**lobal **C**ontext **E**nhanced **G**raph Neural **N**etworks (GCE-GNN) to exploit item transitions over all sessions in a more subtle manner for better inferring the user preference of the current session. Specifically, GCE-GNN learns two levels of item embeddings from *session* graph and *global* graph, respectively: (i) *Session graph*, which is to learn the session-level item embedding by modeling pairwise item-transitions within the current session; and (ii) *Global graph*, which is to learn the global-level item embedding by modeling pairwise item-transitions over all sessions. In GCE-GNN, we propose a novel *global-level item representation learning layer*, which employs a session-aware attention mechanism to recursively incorporate the neighbors' embeddings of each node on the *global* graph. We also design a *session-level item representation learning layer*, which employs a GNN on the *session* graph to learn *session*-level item embeddings within the current session. Moreover, GCE-GNN aggregates the learnt item representations in the two levels with a soft attention mechanism. Experiments on three benchmark datasets demonstrate that GCE-GNN outperforms the state-of-the-art methods consistently.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommendation system; Session-based recommendation; Graph neural network

ACM Reference Format:
Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, Minghui Qiu. 2020. Global Context Enhanced Graph Neural Networks for Session-based Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401142

1 INTRODUCTION

Recommendation systems play critical roles on various on-line platforms, due to their success in addressing information overload problem by recommending useful content to users. Conventional recommendation approaches (*e.g.,* collaborative filtering [11]) usually rely on the availability of user profiles and long-term historical interactions, and may perform poorly in many recent real-world scenarios, *e.g., mobile stream media* like YouTube[1] and Tiktok[2], when such information is unavailable (*e.g.,* unlogged-in user) or limited available (*e.g.,* short-term historical interaction). Consequently, session-based recommendation has attracted extensive attention recently, which predicts the next interested item based on a given anonymous behavior sequence in chronological order.

Most of early studies on session-based recommendation fall into two categories, *i.e.,* similarity-based [11] and chain-based [12]. The former heavily replies on the co-occurrence information of items in the current session while neglecting the sequential behavior patterns. The later infers all possible sequences of user choices over all items, which may suffer from intractable computation problem for real-world applications where the number of items is large. Recently, many deep learning based approaches are proposed for the task, which make use of pairwise item-transition information to model the user preference of a given session [2, 4, 6, 18, 19, 21]. These approaches have achieved encouraging results, but they still face the following issues. *First*, some of them infer the anonymous user's preference by sequentially extracting the session's pairwise item-transition information in chronological order using recurrent

⊠: Corresponding Author

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401142

[1] https://www.youtube.com/.
[2] https://www.tiktok.com/.

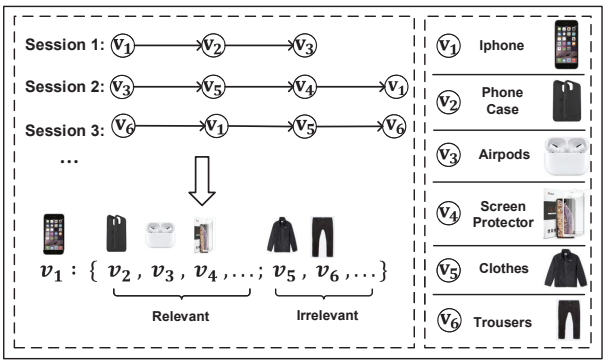


Figure 1: A toy example of *global*-level item transition modeling.

neural networks (RNN) (*e.g.,* GRU4REC [2], NARM [6]) and memory networks (*e.g.,* STAMP [8]). However, a session may contain multiple user choices and even noise, and thus they may be insufficient in generating all correct dependencies, which suffer from the inability of modeling the complicated inherent order of item-transition patterns in embedding. *Second*, the others are based on graph neural networks [7, 24, 25] with self-attention mechanisms such as SR-GNN [24]. They learn the representation of the entire session by calculating the relative importance based on the session's pairwise item-transition between each item and the last one, and the performance heavily rely on the relevance of the last item to the user preference of the current session.

Furthermore, almost all the previous studies model user preference only based on the current session while ignoring the useful item-transition patterns from other sessions. To the best of our knowledge, CSRM [19] is the only work incorporating collaborative information from the latest m sessions to enrich the representation of the current session in end-to-end manner. CSRM treats sessions as the minimum granularity and measure similarities between the current and the latest m sessions to extract collaborative information. However, it may unfortunately encode both relevant and irrelevant information of the other sessions into the current session embeddings, which may even deteriorate the performance [21]. We illustrate this with an example in Figure 1. Without loss of generality, suppose the current session is "Session 2", and the session-based recommendation aims to recommend the relevant accessories related to "Iphone ". From Figure 3, we observe that: (i) Utilizing the item-transition of the other sessions might help model the user preference of the current session. For example, we can find *relevant* pairwise item-transition information for Session 2 from "Session 1" and "Session 3", *e.g.,* a new pairwise item-transition "[Iphone, Phone Case]"; and (ii) Directly utilizing the item-transition information of the *entire* other session may introduce noise when part of the item-transition information encoded in such session is not relevant to the current session. For instance, CSRM [19] may also consider to utilize "Session 3" to help modeling the user preference of "Session 2" if "Session 3" is one of the latest m sessions, and it will introduce the *irrelevant* items (*i.e.,* "clothes" and "trousers") when learning "Session 2"'s embedding as it treats "Session 3" as a whole without distinguishing relevant item-transition from irrelevant item-transition, which is challenging.

To this end, we propose a novel approach to exploit the item-transitions over all sessions in a more subtle manner for better inferring the user preference of the current session for session-based recommendation, which is named **G**lobal **C**ontext **E**nhanced **G**raph **N**eural **N**etworks (GCE-GNN). In GCE-GNN, we propose to learn two levels of item embeddings from *session* graph and *global* graph, respectively: (i) *Session graph*, which is to learn the session-level item embedding by modeling pairwise item-transitions within the current session; and (ii) *Global graph*, which is to learn the global-level item embeddings by modeling pairwise item-transitions over sessions (including the current session). In GCE-GNN, we propose a novel *global-level item representation learning layer*, which employs a session-aware attention mechanism to recursively incorporate the neighbors' embeddings of each node on the *global* graph. We also design a *session-level item representation learning layer*, which employs a GNN on the *session* graph to learn *session*-level item embeddings within the current session. Moreover, GCE-GNN aggregates the learnt item representations in the two levels with a soft attention mechanism.

The main contributions of this work are summarized as follows:

- To the best of our knowledge, this is the first work of exploiting *global*-level item-transitions over all sessions to learn *global*-level contextual information for session-based recommendation.
- We propose a unified model to improve the recommendation performance of the current session by effectively leveraging the pairwise item-transition information from two levels of graph models, *i.e., session* graph and *global* graph.
- We also propose a position-aware attention to incorporate the reversed position information in item embedding, which shows the superiority performance for session-based recommendation.
- We conduct extensive experiments on three real-world datasets, which demonstrate that GCE-GNN outperforms nine baselines including state-of-the-art methods.

2 RELATED WORK

Markov Chain-based SBR. Several traditional methods can be employed for SBR although they are not originally designed for SBR. For example, markov Chain-based methods map the current session into a Markov chain, and then infer a user's next action based on the previous one. Rendle *et al.* [10] propose FPMC to capture both sequential patterns and long-term user preference by a hybrid method based on the combination of matrix factorization and first-order Markov chain for recommendation. It can be adapted for SBR by ignoring the user latent representation as it is not available for anonymous SBR. However, MC-based methods usually focus on modeling sequential transition of two adjacent items. In contrast, our proposed model converts the sequentially item-transitions into graph-structure data for capturing the inherent order of item-transition patterns for SBR.

Deep-learning based SBR. In recent years, neural network-based methods that are capable of modeling sequential data have been

utilized for SBR. Hidasi *et al.* [2] propose the first work called GRU4REC to apply the RNN networks for SBR, which adopts a multi-layer Gated Recurrent Unit (GRU) to model item interaction sequences. Then, Tan *et al.* [15] extend the method [2] by introducing data augmentation. Li *et al.* [6] propose NARM that incorporates attention mechanism into stack GRU encoder to capture the more representative item-transition information for SBR. Liu *et al.* [8] propose an attention-based short-term memory networks (named STAMP) to captures the user's current interest without using RNN. Both NARM and STAMP emphasize the importance of the last click by using attention mechanism. Inspired by $Transformer$ [16], SASRec [4] stacks multiple layers to capture the relevance between items. ISLF [13] takes into account the user's interest shift, and employs variational auto-encoder (VAE) and RNN to capture the user's sequential behavior characteristics for SBR. MCPRN [21] proposes to model the multi-purpose of a given session by using a mixture-channel model for SBR. However, similar to MC-based methods, RNN-based methods focus on modeling the sequential transitions of adjacent items [20] to infer user preference via the chronology of the given sequence, and thus cannot model the complex item-transition patterns (e.g., non-adjacent item transitions).

Recently, several proposals employ GNN-based model on graph built from the current session to learn item embeddings for SBR. Wu *et al.* [24] propose a gated GNN model (named SR-GNN) to learn item embeddings on the session graph, and then obtain a representative session embedding by integrating each learnt item embedding with attentions, which is calculated according to the relevance of each item to the last one. Following the success of SR-GNN, some variants are also proposed for SBR, such as GC-SAN [25]. Qiu *et al.* [9] propose FGNN to learn each item representation by aggregating its neighbors' embeddings with multi-head attention, and generate the final session representation by repeatedly combining each learnt embeddings with the relevance of each time to the session. However, all these approaches only model the item-transition information on the current session. In contrast, our proposed model learns the item-transition information over all sessions to enhance the learning from the current session.

Collaborative Filtering-based SBR. Although deep learning based methods have achieved remarkable performance, collaborative filtering (CF) based methods can still provide competitive results. Item-KNN [11] can be extended for SBR by recommending items that are most similar to the last item of the current session. KNN-RNN [3] makes use of GRU4REC [2] and the co-occurrence-based KNN model to extract the sequential patterns for SBR. Recently, Wang *et al.* [19] propose an end-to-end neural model named CSRM, which achieves state-of-the-art performance. It first utilizes NARM over item-transitions to encode each session, then enriches the representation of the current session by exploring the latest m neighborhood sessions, and finally utilizes a fusion gating mechanism to learn to combine different sources of features. However, it may suffer from noise when integrating other sessions' embeddings for the current one. In contrast, our proposed method considers the collaborative information in *item-level*: we use the item embeddings in other sessions to enrich the item embeddings of the current session, and then integrate them into session representation for SBR.

3 PRELIMINARIES

In this section, we first present the problem statement, and then introduce two types of graph models, *i.e., session* graph and *global* graph, based on different levels of pair-wise item transitions over sessions for learning item representations, in which we highlight the modeling of *global*-level item transition information as it is the basis of *global* graph construction.

3.1 Problem Statement

Let $V = \{v_1, v_2, ..., v_m\}$ be all of items. Each anonymous session, which is denoted by $S = \{v_1^s, v_2^s, ..., v_l^s\}$, consists of a sequence of interactions (*i.e.,* items clicked by a user) in chronological order, where v_i^s denotes item v_i clicked within session S, and the length of S is l.

Given a session S, the problem of session-based recommendation aims to recommend the top-N items ($1 \leq N \leq |V|$) from V that are most likely to be clicked by the user of the current session S.

3.2 Graph Models: Session Graph and Global Graph

In this subsection, we present two different graph models to capture different levels of *item transition* information over all available sessions for item representation learning.

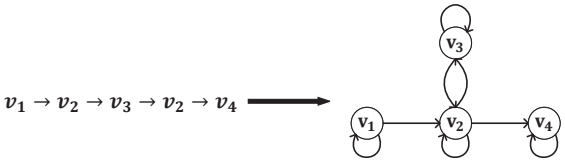
3.2.1 Session Graph Model. Session-based graph aims to learn the ***session-level*** item embedding by modeling sequential patterns over pair-wise adjacent items in the current session. Inspired by [24], each session sequence is converted into a session graph for learning the embeddings of items in the current session via GNN, which is defined as follows, given session $S = \{v_1^s, v_2^s, ..., v_l^s\}$, let $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ be the corresponding *session* graph, where $\mathcal{V}_s \subseteq V$ is the set of clicked items in S, $\mathcal{E}_s = \{e_{ij}^s\}$ denotes the edge set, in which each edge indicates two adjacent items (v_i^s, v_j^s) in S, which is called ***session***-level item-transition pattern. By following the work [9], each item is added a self loop (*rf.* Figure 2a).

Different from [9, 24], our session graph has four types of edges depending on the relationship between item i and item j which are denoted by r_{in}, r_{out}, r_{in-out} and r_{self}. For edge (v_i^s, v_j^s), r_{in} indicates there is only transition from v_j^s to v_i^s, r_{out} implies there is only transition from v_i^s to v_j^s, and r_{in-out} reveals there are both transitions from v_j^s to v_i^s and from v_i^s to v_j^s; r_{self} refers to the self transition of an item.

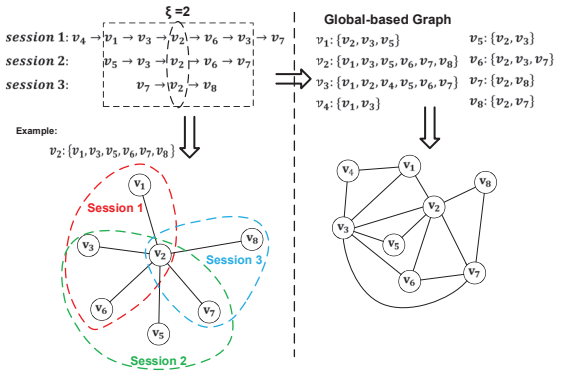
3.2.2 Global Graph Model. Compared with traditional deep learning-based approaches (*e.g.,* RNN-based [6]) that focus on modeling sequential patterns of the entire session, *session* graph can efficiently capture complicated graph patterns of a session to learn *session*-level item embeddings.

However, we also aim to capture item-transition information from other sessions for learning representations of items, which is called ***global*-level item transition information**.

Global-level Item Transition Modeling. Here, we take into account *global*-level item transitions for *global*-level item representation learning, via integrating all pairwise item transitions over sessions. As such, we propose a novel *global* graph model for learning *global*-level item embeddings, which breaks down sequence



(a) *Session* Graph.



(b) *Global* Graph.

Figure 2: Illustrations of construction of *session* graph and *global* graph.

independence assumption with linking all pairs of items based on pairwise transitions over all sessions (including the current one). Next, we firstly present a concept (*i.e.,* ε-neighbor set) for modeling global-level item transition, and then give the definition of *global* graph.

Definition 1. ***ε-Neighbor Set*** *($\mathcal{N}_\varepsilon(v)$).* For any item v_i^p in session S_p, the ε-neighbor set of v_i^p indicates a set of items, each element of which is defined as follows,

$$\mathcal{N}_\varepsilon(v_i^p) = \left\{ v_j^q | v_i^p = v_{i'}^q \in S_p \cap S_q; v_j^p \in S_q; j \in [i' - \varepsilon, i' + \varepsilon]; S_p \neq S_q \right\},$$

where i' is the order of item v_i^p in session S_q, ε is a hyperparameter to control the scope of modeling of item-transition between v_i^p and the items in S_q. Note that, parameter ε favors the modeling of short-range item transitions over sessions, since it is helpless (even noise, *e.g.,* irrelevant dependence) for capturing the *global*-level item transition information if beyond the scope (ε).

According to Definition 1, for each item $v_i \in V$, ***global*-level item transition** is defined as $\{(v_i, v_j) | v_i, v_j \in V; v_j \in \mathcal{N}_\varepsilon(v_i)\}$. Notably, we do not distinguish the direction of *global*-level item transition information for efficiency.

Global Graph. Global graph aims to capture the *global*-level item transition information, which will be used to learn item embeddings over all sessions. Specifically, the *global* graph is built based on ε-neighbor sets of items in all sessions. Without loss of generality, *global* graph is defined as follows, let $\mathcal{G}_g = (\mathcal{V}_g, \mathcal{E}_g)$ be the ***global*** graph, where $\mathcal{V}_g$ denotes the graph node set that contains all items in V, and $\mathcal{E}_g = \{e_{ij}^g | (v_i, v_j) | v_i \in V, v_j \in \mathcal{N}_\varepsilon(v_i)\}$ indicates the set of edges, each corresponding to two pairwise items from all the sessions. Figure 2b shows an example of constructing the *global* graph ($\varepsilon = 2$). Additionally, for each node v_i, we generate weight for its adjacent edges to distinguish the importance of v_i's neighbors as follows: For each edge (v_i, v_j) ($v_j \in \mathcal{N}_{v_i}^g$), we use its frequency over all the sessions as its weight of the corresponding edge; we only keep top-N edges with the highest weights for each item v_i on graph $\mathcal{G}_g$ due to efficiency consideration. Note that the definition of the neighbors[3] (*i.e.,* $\mathcal{N}_v^g$) of item v on graph $\mathcal{G}_g$ is same as $\mathcal{N}_\varepsilon(v)$. Hence, $\mathcal{G}_g$ is an *undirected weighted* graph as ε-neighbor set is undirected. During the testing phase, we do not *dynamically* update the topological structure of *global* graph for efficiency consideration.

Remark. Each item in V is encoded into an unified embedding space at time-step t, *i.e.,* $\mathbf{h}_i^t \in \mathbb{R}^d$ (d indicates the dimension of item embedding), which is feed with an initialization embedding $\mathbf{h}_i^0 \in \mathbb{R}^{|V|}$, here we use *one-hot* based embedding and it is transformed into d-dimensional latent vector space by using a trainable matrix $\mathbf{W}_0 \in \mathbb{R}^{d \times |V|}$.

4 THE PROPOSED METHOD

We propose a novel **G**lobal **C**ontext **E**nhanced **G**raph **N**eural **N**etworks for Session-based Recommendation (GCE-GNN). GCE-GNN aims to exploit both *session*-level and *global*-level pairwise item transitions for modeling the user preference of the current session for recommendation. Figure 3 presents the architecture of GCE-GNN, which comprises four main components: 1) *global-level item representation learning layer*. It learn *global*-level item embeddings over all sessions by employing a session-aware attention mechanism to recursively incorporate each node's neighbors' embeddings based on the global graph ($\mathcal{G}_g$) structure; 2) *session-level item representation learning layer*. It employs a GNN model on session graph $\mathcal{G}_s$ to learn *session*-level item embeddings within the current session; 3) *session representation learning layer* It models the user preference of the current session by aggregating the learnt item representations in both session-level and global-level; 4) *prediction layer*. It outputs the predicted probability of candidate items for recommendation. We next present the four components in detail.

4.1 Global-level Item Representation Learning Layer

We next present how to propagate features on *global* graph to encode item-transition information from other sessions to help recommendation.

Our layers are built based on the architecture of graph convolution network [5], and we generate the attention weights based on the importance of each connection by exploiting the idea of graph attention network [17]. Here, we first describe a single layer, which consists of two components: *information propagation* and *information aggregation*, and then show how to generalize it to multiple layers.

Information Propagation: An item may be involved in multiple sessions, from which we can obtain useful item-transition information to effectively help current predictions.

[3]We do not distinguish $\mathcal{N}_\varepsilon(v)$ and $\mathcal{N}_v^g$ when the context is clear and discriminative.

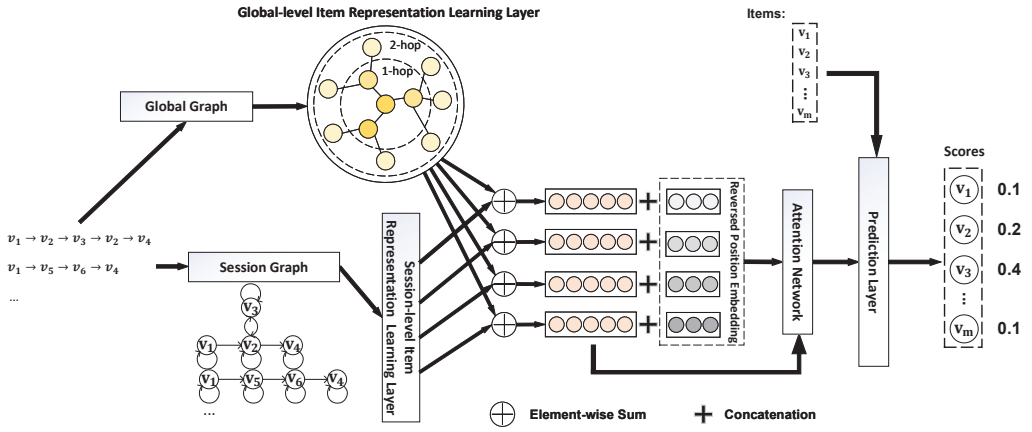


Figure 3: An overview of the proposed framework. Firstly, a global graph is constructed based on all training session sequences. Then for each session, a global feature encoder and local feature encoder will be used to extract node feature with global context and local context. Then the model incorporates position information to learn the contribution of each item to the next predicted item. Finally, candidate items will be scored.

To obtain the first-order neighbor's features of item v, one straightforward solution is to use mean pooling method [1]. However, not all of items in v's ε-neighbor set are relevant to the user preference of the current session, and thus we consider to utilize a session-aware attention to distinguish the importance of items in $(\mathcal{N}_\varepsilon(v))$. Therefore, each item in $\mathcal{N}_\varepsilon(v)$ is linearly combined according to the session-aware attention score,

$$\mathbf{h}_{\mathcal{N}_{v_i}^g} = \sum_{v_j \in \mathcal{N}_{v_i}^g} \pi(v_i, v_j)\mathbf{h}_{v_j}, \tag{1}$$

where $\pi(v_i, v_j)$ estimates the importance weight of different neighbors. Intuitively, the closer an item is to the preference of the current session, the more important this item is to the recommendation. Therefore we implement $\pi(v_i, v_j)$ as follows:

$$\pi(v_i, v_j) = \mathbf{q}_1^T \text{LeakyRelu}\big(\mathbf{W}_1[(\mathbf{s} \odot \mathbf{h}_{v_j}) \| w_{ij}]\big), \tag{2}$$

here we choose LeakyRelu as activation function, $\odot$ indicates element-wise product, $\|$ indicates concatenation operation, $w_{ij} \in \mathbb{R}^1$ is the weight of edge (v_i, v_j) in global graph, $\mathbf{W}_1 \in \mathbb{R}^{d+1 \times d+1}$ and $\mathbf{q}_1 \in \mathbb{R}^{d+1}$ are trainable parameters, $\mathbf{s}$ can be seen as the features of current session, which is obtained by computing the average of item representations of the current session,

$$\mathbf{s} = \frac{1}{|S|} \sum_{v_i \in S} \mathbf{h}_{v_i}. \tag{3}$$

Distinct from mean pooling, our approach makes the propagation of information dependent on the affinity between S and v_j, which means neighbors that match the preference of current session will be more favourable.

Then we normalize the coefficients across all neighbors connected with v_i by adopting the softmax function:

$$\pi(v_i, v_j) = \frac{\exp\big(\pi(v_i, v_j)\big)}{\sum_{v_k \in \mathcal{N}_{v_i}^g} \exp\big(\pi(v_i, v_k)\big)}. \tag{4}$$

As a result, the final attention score is capable of suggesting which neighbor nodes should be given more attention.

Information Aggregation: The final step is to aggregate the item representation $\mathbf{h}_v$ and its neighborhood representation $h_{\mathcal{N}_v}^g$, we implement aggregator function agg as follows,

$$\mathbf{h}_v^g = \text{relu}\big(\mathbf{W}_2[\mathbf{h}_v \| \mathbf{h}_{\mathcal{N}_v^g}]\big), \tag{5}$$

where we choose relu as the activation function and $\mathbf{W}_2 \in \mathbb{R}^{d \times 2d}$ is transformation weight.

Through a single aggregator layer, the representation of an item is dependent on itself and its immediate neighbors. We could explore the high-order connectivity information through extending aggregator from one layer to multiple layers, which allows more information related to the current session to be incorporated into the current representation. We formulate the representation of an item in the k-th steps as:

$$\mathbf{h}_v^{g,(k)} = \text{agg}\big(\mathbf{h}_v^{(k-1)}, \mathbf{h}_{\mathcal{N}_v^g}^{(k-1)}\big), \tag{6}$$

$\mathbf{h}_v^{(k-1)}$ is representation of item v which is generated from previous information propagation steps, $\mathbf{h}_v^{(0)}$ is set as $\mathbf{h}_v$ at the initial propagation iteration. In this way, the k-order representation of an item is a mixture of its initial representations and its neighbors up to k hops away. This enables more effective messages to be incorporated into the representation of the current session.

4.2 Session-level Item Representation Learning layer

The session graph contains pairwise item-transitions within the current session. We next present how to learn the session-level item embedding.

As the neighbors of item in session graph have different importance to itself, we utilize attention mechanism to learn the weight between different nodes. The attention coefficients can be computed through *element-wise product* and *non-linear transformation*:

$$e_{ij} = \text{LeakyReLU}\left(\mathbf{a}_{r_{ij}}^\top \left(\mathbf{h}_{v_i} \odot \mathbf{h}_{v_j}\right)\right), \tag{7}$$

where e_{ij} indicates the importance of node v_j's features to node v_i and we choose LeakyReLU as activation function, r_{ij} is the relation between v_i and v_j and $\mathbf{a}_* \in \mathbb{R}^d$ are weight vectors.

For different relations, we train four weight vectors, namely a_{in}, a_{out}, a_{in-out} and a_{self}. As not every two nodes are connected in the graph, we only compute e_{ij} for nodes $j \in \mathcal{N}_{v_i}^s$ to inject the graph structure into the model, where $\mathcal{N}_{v_i}^s$ is the first-order neighbors of v_i. And to make coefficients comparable across different nodes, we normalize the attention weights through softmax function:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}_{r_{ij}}^\top \left(\mathbf{h}_{v_i} \odot \mathbf{h}_{v_j}\right)\right)\right)}{\sum_{v_k \in \mathcal{N}_{v_i}^s} \exp\left(\text{LeakyReLU}\left(\mathbf{a}_{r_{ik}}^\top \left(\mathbf{h}_{v_i} \odot \mathbf{h}_{v_k}\right)\right)\right)}. \tag{8}$$

In Eq. (8) the attention coefficients α_{ij} is asymmetric, as their neighbors are different, which means the contribution they make to each other are unequal. Next we obtain the output features for each node by computing a linear combination of the features corresponding to the coefficients:

$$\mathbf{h}_{v_i}^s = \sum_{v_j \in \mathcal{N}_{v_i}^s} \alpha_{ij} \mathbf{h}_{v_j}. \tag{9}$$

The item representations in session graph is aggregated by the features of item itself and its neighbor in the current session. Through the attention mechanism, the impact of noise on the session-level item representation learning is reduced.

4.3 Session Representation Learning Layer

For each item, we obtain its representations by incorporating both global context and session context, and its final representation is computed by sum pooling,

$$\begin{aligned} \mathbf{h}_v^{g,(k)} &= \text{dropout}\left(\mathbf{h}_v^{g,(k)}\right) \\ \mathbf{h}'_v &= \mathbf{h}_v^{g,(k)} + \mathbf{h}_v^s, \end{aligned} \tag{10}$$

here we utilize dropout[14] on global-level representation to avoid overfitting.

Based on the learnt item representations, we now present how to obtain the session representations. Different from previous work [8, 24, 25] which mainly focus on the last item, in this paper we propose a more comprehensive strategy to learn the contribution of each part of the session for prediction.

In our method, a session representation is constructed based on all the items involved in the session. Note that the contribution of different items to the next prediction is not equal. Intuitively, the items clicked later in the session are more representative of the user's current preferences, which shows their greater importance for the recommendation. Moreover, it is important to find the main purpose of the user and filter noise in current session [6]. Hence we incorporate reversed position information and session information to make a better prediction.

After feeding a session sequence into graph neural networks, we can obtain the representation of the items involved in the session, *i.e.,* $\mathbf{H} = \left[\mathbf{h}'_{v_1^s}, \mathbf{h}'_{v_2^s}, ..., \mathbf{h}'_{v_l^s}\right]$. We also use a learnable position embedding matrix $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_l]$, where $\mathbf{p}_i \in \mathbb{R}^d$ is a position vector for specific position i and l is the length of the current session sequence. The position information is integrated through *concatenation* and *non-linear transformation*:

$$\mathbf{z}_i = \tanh\left(\mathbf{W}_3 \left[\mathbf{h}'_{v_i^s} \parallel \mathbf{p}_{l-i+1}\right] + \mathbf{b}_3\right), \tag{11}$$

where parameters $\mathbf{W}_3 \in \mathbb{R}^{d \times 2d}$ and $\mathbf{b}_3 \in \mathbb{R}^d$ are trainable parameters. Here we choose the reversed position embedding because the length of the session sequence is not fixed. Comparing to forward position information, the distance of the current item from the predicted item contains more effective information, *e.g.,* in the session $\{v_2 \rightarrow v_3 \rightarrow ?\}$, v_3 is the second in the sequence and shows great influence to prediction, however in the session $\{v_2 \rightarrow v_3 \rightarrow v_5 \rightarrow v_6 \rightarrow v_8 \rightarrow ?\}$, the importance of v_3 would be relatively small. Therefore the reversed position information can more accurately suggest the importance of each item.

The session information is obtained by computing the average of item representations of the session,

$$\mathbf{s}' = \frac{1}{l} \sum_{i=1}^{l} \mathbf{h}'_{v_i^s}. \tag{12}$$

Next, we learn the corresponding weights through a soft-attention mechanism:

$$\beta_i = \mathbf{q}_2^\top \sigma\left(\mathbf{W}_4 \mathbf{z}_i + \mathbf{W}_5 \mathbf{s}' + \mathbf{b}_4\right), \tag{13}$$

where $\mathbf{W}_4, \mathbf{W}_5 \in \mathbb{R}^{d \times d}$ and $\mathbf{q}_2, \mathbf{b}_4 \in \mathbb{R}^d$ are learnable parameters.

Finally, the session representation can be obtained by linearly combining the item representations:

$$\mathbf{S} = \sum_{i=1}^{l} \beta_i \mathbf{h}'_{v_i^s}. \tag{14}$$

The session representation $\mathbf{S}$ is constructed by all the items involved in the current session, where the contribution of each item is determined not only by the information in the session graph, but also by the chronological order in the sequence.

4.4 Prediction Layer

Based on the obtained session representations $\mathbf{S}$, the final recommendation probability for each candidate item based on their initial embeddings as well as current session representation, and we first use dot product and then apply softmax function to obtain the output $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}}_i = \text{Softmax}\left(\mathbf{S}^\top \mathbf{h}_{v_i}\right), \tag{15}$$

where $\hat{\mathbf{y}}_i \in \hat{\mathbf{y}}$ denotes the probability of item v_i appearing as the next-click in the current session.

The loss function is defined as the cross-entropy of the prediction results $\hat{\mathbf{y}}$:

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{m} \mathbf{y}_i \log\left(\hat{\mathbf{y}}_i\right) + \left(1 - \mathbf{y}_i\right) \log\left(1 - \hat{\mathbf{y}}_i\right), \tag{16}$$

where $\mathbf{y}$ denotes the one-hot encoding vector of the ground truth item.

5 EXPERIMENTS

We have conducted extensive experiments to evaluate the accuracy of the proposed GCE-GNN method by answering the following five key research questions:

- **RQ1**: Does GCE-GNN outperform state-of-the-art SBR baselines in real world datasets?

Table 1: Statistics of the used datasets.

Dataset	Diginetica	Tmall	Nowplaying
# click	982,961	818,479	1,367,963
# train	719,470	351,268	825,304
# test	60,858	25,898	89,824
# items	43,097	40,728	60,417
avg. len.	5.12	6.69	7.42

- **RQ2**: Does global graph and global-level encoder improve the performance of GCE-GNN? How well does GCE-GNN perform with different depth of receptive field k?
- **RQ3**: Is reversed position embedding useful?
- **RQ4**: How well does GCE-GNN perform with different aggregation operations?
- **RQ5**: How do different hyper-parameter settings (*e.g.,* node dropout) affect the GCE-GNN's accuracy?

5.1 Datesets and Preprocessing

We employ three benchmark datasets, namely, *Diginetica*[4], *Tmall*[5] and *Nowplaying*[6]. Particularly, Diginetica dataset is from CIKM Cup 2016, consisting of typical transaction data. Tmall dataset comes from IJCAI-15 competition, which contains anonymized user's shopping logs on Tmall online shopping platform. Nowplaying dataset comes from [26], which describes the music listening behavior of users.

Following [24, 25], we conduct preprocessing step over the three datasets. More specifically, sessions of length 1 and items appearing less than 5 times were filtered across all the three datasets. Similar to [8], we set the sessions of last week (latest data) as the test data, and the remaining historical data for training. Furthermore, for a session $S = [s_1, s_2, ..., s_n]$, we generate sequences and corresponding labels by a sequence splitting preprocessing, i.e., $([s_1], s_2)$, $([s_1, s_2], s_3)$, ..., $([s_1, s_2, ..., s_{n-1}], s_n)$ for both training and testing across all the three datasets. The statistics of datasets, after preprocessing, are summarized in Table 1.

5.2 Evaluation Metrics

We adopt two widely used ranking based metrics: **P@N** and **MRR@N** by following previous work[8, 24].

5.3 Baseline Algorithms

We compare our method with classic methods as well as state-of-the-art models. The following nine baseline models are evaluated.

POP: It recommends top-N frequent items of the training set.

Item-KNN[11]: It recommends items based on the similarity between items of the current session and items of other ones.

FPMC[10]: It combines the matrix factorization and the first-order Markov chain for capturing both sequential effects and user preferences. By following the previous work, we also ignore the user latent representations when computing recommendation scores.

GRU4Rec[7] [2]: It is RNN-based model that uses Gated Recurrent Unit (GRU) to model user sequences.

NARM[8] [6]: It improves over **GRU4Rec**[2] by incorporating attentions into RNN for SBR.

STAMP[9] [8]: It employs attention layers to replace all RNN encoders in previous work by fully relying on the self-attention of the last item in the current session to capture the user's short-term interest.

SR-GNN[10] [24]: It employs a gated GNN layer to obtain item embeddings, followed by a self-attention of the last item as STAMP[8] does to compute the session level embeddings for session-based recommendation.

CSRM[11] [19]: It utilizes the memory networks to investigate the latest m sessions for better predicting the intent of the current session.

FGNN[12] [9]: It is recently proposed by designing a weighted attention graph layer to learn items embeddings, and the sessions for the next item recommendation are learnt by a graph level feature extractor.

5.4 Parameter Setup

Following previous methods [6][8][24], the dimension of the latent vectors is fixed to 100, and the size for mini-batch is set to 100 for all models. We keep the hyper-parameters of each model consistent for a fair comparison. For CSRM, we set the memory size to 100 which is consistent with the batch size. For FGNN, we set the number of GNN layer to 3 and the number of heads is set to 8. For our model, all parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. We use the Adam optimizer with the initial learning rate 0.001, which will decay by 0.1 after every 3 epoch. The L2 penalty is set to 10^{-5} and the dropout ratio is searched in $\{0.1, 0.2, ..., 0.9\}$ on a validation set which is a random 10% subset of the training set. Moreover, we set the number of neighbors and the maximum distance of adjacent items ε to 12 and 3, respectively.

5.5 Overall Comparison (RQ1)

Table 2 reports the experimental results of the 9 baselines and our proposed model on three real-world datasets, in which the best result of each column is highlighted in boldface. It can be observed that GCE-GNN achieves the best performance (statistically significant) across all three datasets in terms of the two metrics (with N=10, and 20) *consistently*, which ascertains the effectiveness of our proposed method.

Among the traditional methods, POP's performance is the worst, as it only recommends top-N frequent items. Comparing with POP, FPMC shows its effectiveness on three datasets, which utilizes first-order Markov chains and matrix factorization. Item-KNN achieves the best results among the traditional methods on the Diginetica and

[4]https://competitions.codalab.org/competitions/11161
[5]https://tianchi.aliyun.com/dataset/dataDetail?dataId=42
[6]http://dbis-nowplaying.uibk.ac.at/#nowplaying
[7]https://github.com/hidasib/GRU4Rec
[8]https://github.com/lijingsdu/sessionRec_NARM
[9]https://github.com/uestcnlp/STAMP
[10]https://github.com/CRIPAC-DIG/SR-GNN
[11]https://github.com/wmeirui/CSRM_SIGIR2019
[12]https://github.com/RuihongQiu/FGNN

Table 2: Effectiveness comparison on three datasets.

Dataset	Diginetica				Tmall				Nowplaying			
Methods	**P@10**	**P@20**	**MRR@10**	**MRR@20**	**P@10**	**P@20**	**MRR@10**	**MRR@20**	**P@10**	**P@20**	**MRR@10**	**MRR@20**
POP	0.76	1.18	0.26	0.28	1.67	2.00	0.88	0.90	1.86	2.28	0.83	0.86
Item-KNN	25.07	35.75	10.77	11.57	6.65	9.15	3.11	3.31	10.96	15.94	4.55	4.91
FPMC	15.43	22.14	6.20	6.66	13.10	16.06	7.12	7.32	5.28	7.36	2.68	2.82
GRU4Rec	17.93	30.79	7.73	8.22	9.47	10.93	5.78	5.89	6.74	7.92	4.40	4.48
NARM	35.44	48.32	15.13	16.00	19.17	23.30	10.42	10.70	13.6	18.59	6.62	6.93
STAMP	33.98	46.62	14.26	15.13	22.63	26.47	13.12	13.36	13.22	17.66	6.57	6.88
CSRM	36.59	50.55	15.41	16.38	24.54	29.46	13.62	13.96	13.20	18.14	6.08	6.42
SR-GNN	38.42	51.26	16.89	17.78	23.41	27.57	13.45	13.72	14.17	18.87	7.15	7.47
FGNN	37.72	50.58	15.95	16.84	20.67	25.24	10.07	10.39	13.89	18.78	6.8	7.15
FGNN(reported)[1]	-	51.36	-	18.47	-	-	-	-	-	-	-	-
GCE-GNN	**41.16**	**54.22**	**18.15**	**19.04**	**28.01**	**33.42**	**15.08**	**15.42**	**16.94**	**22.37**	**8.03**	**8.40**
p-value	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.01

[1] The codes of FGNN model released by the author are incomplete. For fairness, we compare our method with our re-implemented FGNN model as well as the results reported in original paper.

Nowplaying datasets. Note it only applies the similarity between items and does not consider the chronological order of the items in a session, and thus it cannot capture the sequential transitions between items.

Compared with traditional methods, neural network based methods usually have better performance for session-based recommendation. In sprite of preforming worse than Item-KNN on Diginetica, GRU4Rec, as the first RNN based method for SBR, still demonstrates the capability of RNN in modeling sequences. However, RNN is designed for sequence modeling, and session based recommendation problems are not merely a sequence modeling task because the user's preference may change within the session.

The subsequent methods, NARM and STAMP outperform GRU4REC significantly. NARM combines RNN and attention mechanism, which uses the last hidden state of RNN as the main preference of user, this result indicates that directly using RNN to encode the session sequence may not be sufficient for SBR as RNN only models one way item-transition between adjacent items in a session. We also observe that STAMP, a complete attention-based method, achieves better performance than NARM on Tmall, which incorporates a self-attention over the last item of a session to model the short-term interest, this result demonstrates the effectiveness of assigning different attention weights on different items for session encoding. Compared with RNN, attention mechanism appears to be a better option, although STAMP neglects the chronological order of items in a session.

CSRM performs better than NARM and STAMP on Diginetica and Tmall. It shows the effectiveness of using item transitions from other sessions, and also shows the shortcomings of the memory networks used by CSRM that have limited slots, additionally CSRM treats other sessions as a whole one without distinguishing the relevant item-transitions from the irrelevant ones encoded in other sessions.

Among all the baseline methods, the GNN-based methods perform better on the Diginetica and Nowplaying datasets. By modeling every session sequence as a subgraph and applying GNN to encode items, SR-GNN and FGNN demonstrate the effectiveness of applying GNN in session-based recommendation. This indicates that the graph modeling would be more suitable than the sequence modeling, RNN, or a set modeling, the attention modeling for SBR.

Our approach GCE-GNN outperforms SR-GNN and FGNN on all the three datasets. Specifically, GCE-GNN outperforms the SR-GNN by 6.86% on Diginetica, 16.34% on Tmall and 15.71% on Nowplaying on average. Different from SR-GNN and FGNN, our approach integrates information from global context, i.e., other session, and local context, i.e., the current session, and also incorporates relative position information, leading to consistent better performance.

5.6 Impact of Global Feature Encoder (RQ2)

We next conduct experiments on three datasets to evaluate the effectiveness of global-level feature encoder and session-level feature encoder. Specially, we design four contrast models:

- GCE-GNN w/o global: GCE-GNN without global-level feature encoder and only with local feature
- GCE-GNN w/o session: GCE-GNN without session-level feature encoder and only with global feature
- GCE-GNN-1-hop: GCE-GNN with global-level feature encoder, which sets the number of hop to 1.
- GCE-GNN-2-hop: GCE-GNN with global-level feature encoder, which sets the number of hop to 2.

Table 3 shows the comparison between different contrast models. It is clear that with global-level feature encoder, GCE-GNN achieves better performance. Comparing with GCE-GNN w/o global context, GCE-GNN with 1-hop and 2-hop global-level feature encoder can explore item-transition information from other sessions, which helps the model to make more accurate predictions. It can also be observed that GCE-GNN with 2-hop performs better than GCE-GNN with 1-hop on Diginetica, indicating that high-level exploring might obtain more effective information from global graph. In addition, GCE-GNN with 1-hop performs better than GCE-GNN with 2-hop on Tmall, and this indicates that higher-level exploring might also introduce noise.

Table 3: The performance of contrast models.

Dataset	Diginetica		Tmall		Nowplaying	
Measures	P@20	MRR@20	P@20	MRR@20	P@20	MRR @20
w/o global	54.08	18.76	32.96	14.72	**23.11**	7.55
w/o session	51.46	17.34	32.96	12.41	19.10	8.15
1-hop	54.04	18.90	**33.42**	**15.42**	22.37	**8.40**
2-hop	**54.22**	**19.04**	32.58	14.83	22.45	8.29

Table 4: The performance of contrast models.

Dataset	Diginetica		Tmall		Nowplaying	
Measures	P@20	MRR@20	P@20	MRR@20	P@20	MRR @20
GCE-GNN-NP	50.45	17.65	31.16	14.71	19.42	6.05
GCE-GNN-SA	51.68	17.94	25.80	12.94	21.40	7.18
GCE-GNN	**54.22**	**19.04**	**33.42**	**15.42**	**22.37**	**8.40**

Table 5: Effects of different aggregation operations.

Dataset	Diginetica		Tmall		Nowplaying	
Measures	P@20	MRR@20	P@20	MRR@20	P@20	MRR @20
Gate Mechanism	53.84	18.83	32.80	15.33	**22.47**	7.83
Max Pooling	47.69	16.44	31.87	15.39	19.13	6.71
Concatenation	51.72	17.03	31.55	14.89	19.88	7.93
Sum Pooling	**54.22**	**19.04**	**33.42**	**15.42**	22.37	**8.40**

5.7 Impact of Position Vector (RQ3)

The position vector is used to drive GCE-GNN to learn the contribution of each part in the current session. Although SASRec [4] has injected forward position vector into the model to improve performance, we argue that forward position vector has very limited effect on the SBR task. To verify this and evaluate the effectiveness of using the position vector in a reverse order, which is proposed in GCE-GNN, we design a series of contrast models:

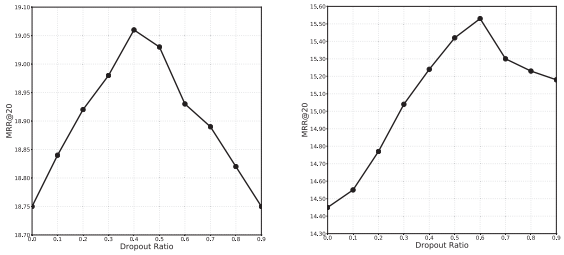
- GCE-GNN-NP: GCE-GNN with forward position vector replacing the reverse order position vector.
- GCE-GNN-SA: GCE-GNN with self attention function replacing the position-aware attention.

Table 4 shows the performance of different contrast models. We observe that our attention network with reversed position embedding performs better than the other two variants.

GCE-GNN-NP does not perform well on all datasets. That is because the model cannot capture the distance from each item to the predicted item, which will mislead the model when training for sessions of various lengths.

GCE-GNN-SA performs better than GCE-GNN-NP on three datasets, indicating that the last item in a session contains the most relevant information for recommendation. However, it does not perform well on Tmall dataset, as it lacks a more comprehensive judgment of the contribution of each item.

Comparing with the two variants, reversed position embedding demonstrates its effectiveness. This confirms that the reversed position information can more accurately suggest the importance of each item. Moreover, though the attention mechanism, we filter the noise in the current session, which makes the model perform better.



(a) Diginetica (b) Tmall

Figure 4: Comparison of overall expert finding performance.

5.8 Impact of Aggregation Operations (RQ4)

As we use local feature encoder and global feature encoder, it is meaningful to compare GCE-GNN with different aggregation operations, *i.e.*,, gating mechanism, max pooling and concatenation mechanism.

For gating mechanism, we use a linear interpolation between local feature representation h^l and global feature representation h^g:

$$\begin{aligned} \mathbf{r}_v &= \sigma(\mathbf{W}_s \mathbf{h}_v^s + \mathbf{W}_g \mathbf{h}_v^g) \\ \mathbf{h}'_v &= \mathbf{r}_v \mathbf{h}_v^g + (\mathbf{1} - \mathbf{r}_v)\mathbf{h}_v^s, \end{aligned} \tag{17}$$

where σ is the sigmoid activation function and $\mathbf{r}_v$ is learned to balance the importance of two features.

For max pooling, we take the maximum value of every dimension for each feature, and the i−th dimension of an item representation $\mathbf{h}'_{vi}$ is formulated as

$$\mathbf{h}'_{vi} = \max(\mathbf{h}_{vi}^g, \mathbf{h}_{vi}^s). \tag{18}$$

For the concatenation operation, the final representation is the concatenation of vectors $\mathbf{h}_v^g$ and $\mathbf{h}_v^s$:

$$\mathbf{h}'_v = \mathbf{M}\big([\mathbf{h}_v^g \| \mathbf{h}_v^s]\big) \tag{19}$$

where $\mathbf{M} \in \mathbb{R}^{d \times 2d}$ is the transformer weight.

Table 5 shows the performance of different aggregation operations on the three datasets. It can be observed that GCE-GNN with sum pooling outperforms other aggregation operations on Diginetica and Tmall in terms of Recall@20 and MRR@20. Max pooling's performance is the worst on Diginetica but it performs better than the other two aggregators on Tmall in terms of MRR@20. Despite of using additional parameters, Gate mechanism and Concatenation's performance is also worse than sum pooling, possibly because too many parameters may lead to overfitting.

5.9 Impact of Dropout Setting (RQ5)

To prevent GCE-GNN from overfitting, we employ dropout [14] regularization techniques, which have been shown to be effective in various neural network architectures including graph neural networks[22][23]. The key idea of dropout is to randomly drop neurons with probability p during training, while using all neurons for testing. Figure 4 shows the impact of dropout in Equation (10) on Diginetica and Tmall datasets. We can observe that when dropout ratio is small, the model does not perform well on both

datasets, as it is easy to overfit. It achieves the best performance when dropout ratio is set to 0.4 on Diginetica and 0.6 on Tmall. However, when dropout ratio is big, the performance of the model starts to deteriorate, as it is hard for the model to learn from data with limited available neurons.

6 CONCLUSION

This paper studies the problem of session-based recommendation, which is a challenging task as the user identities and historical interactions are often unavailable due to privacy and data protection concern. It proposes a novel architecture for session-based recommendation based on graph neural network. Specifically, it first converts the session sequences into session graphs and construct a global graph. The local context information and global context information are subsequently combined to enhance the feature presentations of items. Finally, it incorporates the reversed position vector and session information to empower the proposed model to better learn the contribution of each item. Comprehensive experiments demonstrate that the proposed method significantly outperforms nine baselines over three benchmark datasets consistently.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant No.61602197 and Grant No.61772076, and in part by Equipment Pre-Research Fund for The 13th Five-year Plan under Grant No.41412050801.

REFERENCES

[1] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1024–1034.
[2] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.
[3] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys*. 306–310.
[4] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. 197–206.
[5] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[6] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. 1419–1428.
[7] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. In *ICLR*.
[8] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *SIGKDD*. 1831–1839.
[9] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks. In *CIKM*. 579–588.
[10] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.
[11] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms.. In *WWW*. 285–295.
[12] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *JMLR*, 1265–1295.
[13] Jing Song, Hong Shen, Zijing Ou, Junyi Zhang, Teng Xiao, and Shangsong Liang. 2019. ISLF: Interest Shift and Latent Factors Combination Model for Session-based Recommendation. In *IJCAI*. 5765–5771.
[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* (2014), 1929–1958.
[15] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 17–22.
[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.
[18] Huizhao Wang, Guanfeng Liu, An Liu, Zhixu Li, and Kai Zheng. 2019. DMRAN:A Hierarchical Fine-Grained Attention-Based Network for Recommendation. In *IJCAI*. 3698–3704.
[19] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A Collaborative Session-based Recommendation Approach with Parallel Memory Modules. In *SIGIR*.
[20] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet Orgun. 2019. Sequential Recommender Systems: Challenges, Progress and Prospects. In *IJCAI*. 6332–6338.
[21] Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, Mehmet Orgun, and Longbing Cao. 2019. Modeling Multi-Purpose Sessions for Next-Item Recommendations via Mixture-Channel Purpose Routing Networks. In *IJCAI*. 3771–3777.
[22] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*.
[23] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.
[24] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*. 346–353.
[25] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *IJCAI*. 3940–3946.
[26] Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. 2014. #nowplaying Music Dataset: Extracting Listening Behavior from Twitter. In *ISMM*. 21–26.