Neural Computing and Applications (2022) 34:7581–7596
https://doi.org/10.1007/s00521-021-06859-x

ORIGINAL ARTICLE

IC-GAR: item co-occurrence graph augmented session-based recommendation

Tajuddeen Rabiu Gwadabe[1,2] · Ying Liu[1,2,3]

Received: 23 February 2021 / Accepted: 12 December 2021 / Published online: 8 January 2022

Abstract
Session-based recommendation aims to recommend the next item of an anonymous user session. Previous models consider only the current session and learn both of the user's global and local preferences. These models fail to consider an important source of information, i.e., the co-occurrence pattern of items in different sessions. The co-occurrence patterns elicit the trajectory of other similar users and can improve the recommendation performance. We propose an Item Co-occurrence Graph Augmented Session-based Recommendation (IC-GAR) model, a novel session-based recommendation model that augments the representations of the current session with session co-occurrence patterns. IC-GAR consists of three modules: Encode Module, Session Co-occurrence Module and Prediction Module. The Encoder Module learns both of the user's global and local preference from the current session using Gate Recurrent Units (GRU). The Session Co-occurrence Module uses a modified variant of Graph Convolutional Network (GCN) to model higher order interactions between the item transition patterns in the training sessions. By aggregating the GCN representation of items of the current session, session co-occurrence representation is learned. The Prediction Module decomposes global preference, local preference and session co-occurrence to predict the probability scores of candidate items. Extensive experiments on three publicly available datasets are conducted to demonstrate the effectiveness of IC-GAR. 8.5–39.2% improvement are achieved across datasets in Precision @5, 10 and MRR@5, 10.

Keywords Session-based recommendation · Graph neural networks · Sequential recommendation · Item co-occurrence graph

1 Introduction

Recommender systems aim to predict the items a user might be interested-in based on her previous interactions. Recommender systems have recently become very important, especially in e-commerce due to availability of a large pool of items a user can select from. Recommender systems play a crucial role to consumers and owners of the business. Traditional recommender systems employ Matrix Factorization (MF) [1, 2] methods to learn a low rank user representation from the ratings of previous interactions. By using this representation, the recommender system then predicts the rating of other items the user maybe interested in.

However, on most online platforms, users do not explicitly rate items. Rather, implicit feedback, such as clicks, must be relied on for recommendation. Since user interests are dynamic, traditional MF methods cannot capture such changes. To utilize implicit feedback and capture user interest drifts, researchers have focused on sequential recommender systems. A particular case of sequential recommendation, called session-based recommendation has gained a lot of attention recently. In session-based recommendation, sessions cannot be linked to a

✉ Ying Liu
yingliu@ucas.ac.cn

Tajuddeen Rabiu Gwadabe
tgwadabe@mails.ucas.ac.cn

[1] School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 101400, China

[2] Data Mining and High-Performance Computing Lab, Chinese Academy of Sciences, Beijing 101400, China

[3] Key Lab of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing 100190, China

particular user which may be warranted due to privacy concerns.

To effectively recommend relevant items to users in session-based recommendation, three important criteria have to be considered; short-term preference, long-term preference and session co-occurrence patterns. Consider the example in Fig. 1, in *Session 1* the user watched the "Fast and Furious" movies in serial order from the fifth movie to the seventh movie. Based on the watch history, it will be a good idea to recommend the next movie (eighth movie). In this instance, the last watch (clicked) item is important for recommendation and it is captured by the short-term preference. However, from *Session 2*, the watch history includes animation movies and action movies. The user may not be really interested in the animation movie but watched them with her kids. In this case, recommending another animation movie may not be the best decision but rather recommending both action movie and animation movie. Here the short-term preference is insufficient. However, the long-term preference addresses this issue by capturing the overall session interest. Consider *Session 3*, the user is obviously interest in action movies, however, using the knowledge from *Session 1* that after watching "movie 3" users watch "movie 4", it may be relevant to recommend "movie 4." The session co-occurrence pattern captures the inter-session interaction for improved recommendation.

Short-term preference is represented by the most recent interactions. Markov chain (MC) models have shown to be successful on this task [3, 4]. FMPC [5] assumes independence between interactions and models the first-order MC for sequential recommendation. Older interactions are important to fully understand session long-term preference due to drift in user interest. Here MC models fail due to the independent assumption and the difficulty in the scalability of higher order MC models. Recurrent Neural Networks (RNN) models are a great alternative to MC models for modeling longer sequence and have become the state-of-the-art in session-based recommendation [6–9]. NARM [10], for example, models both user's short-term and long-term preference using GRU with the last hidden state as the short-term user preference. An attention mechanism is then used to learn a user's long-term preference.

Since sessions cannot be tied to particular users, item co-occurrence patterns can elicit behavioral pattern between different sessions. The existing session-based recommendation models consider only the current session for recommendation. However, user behavior can be influenced by others as the old adage goes "*birds of the same feather flock together*". Studies have shown that recommender systems are subjected to conformity bias [11, 12]. That is, users are influenced by the actions of others. Researchers [13, 14] have leverage this trend to improve recommendation. However, in session-based recommendation, user information or social information is not readily available.

To this end, we propose IC-GAR (**I**tem **C**o-occurrence **G**raph **A**ugmented Session-based **R**ecommendation) model that efficiently combines the three important criteria in a session-based recommendation problem. To model the short-term and long-term user preference, we use a GRU. The last hidden state of the GRU represents the user's short-term interest. We then use attention mechanism to capture the long-term interest of users from all hidden states of the GRU. To model item co-occurrence, we first construct a weighted undirected graph containing all the training sessions. Each weighted edge of the graph represents the frequency of transitions from one item to another. By using a variant of Graph Convolutional Network GCN [15], we can learn an item representation that is aware of the various transition patterns between that item and all the other items. We then aggregate the item representations from the GCN for each session to learn the session co-occurrence representation. By using the short-term, the

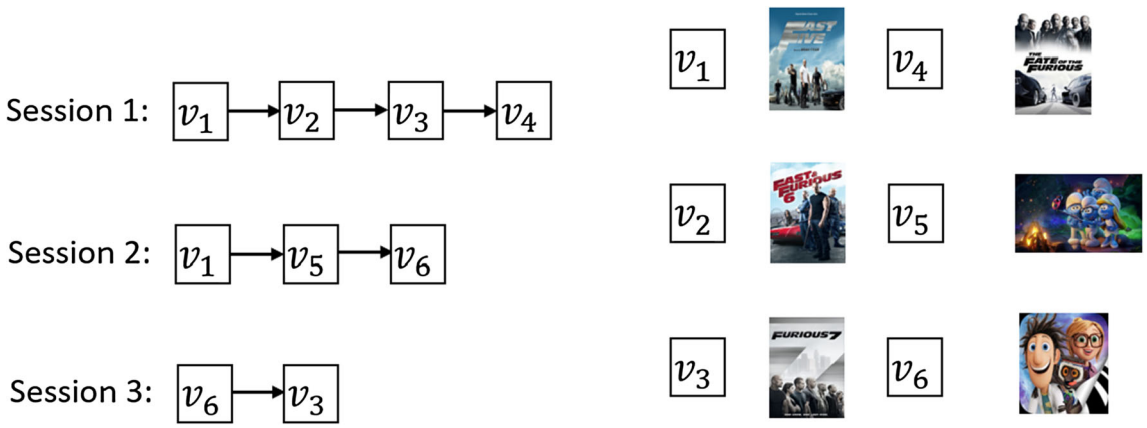

Fig. 1 A toy example of relevance of short-term, long-term and session co-occurrence pattern in session-based recommendation

long-term and the session co-occurrence representations, we then employ a trilinear decomposition to recommend the most relevant items.

In summary, our main contributions are as follows:

- We propose a model that considers three factors i.e., short-term, long-term and session-occurrence representation for session-based recommendation
- A novel IC-GAR model is proposed that accounts for user interest dynamics and item co-occurrence patterns in an end-to-end neural network.
- A graph representation is proposed to learn the session co-occurrence representation in all training sessions.
- We conduct extensive experiments on three datasets to demonstrate the effectiveness of IC-GAR. The proposed IC-GAR model significantly outperforms the state-of-the-art models in terms of MRR and Precision.

The rest of the paper is organized as follows: Sect. 2 presents related works, while Sect. 3 gives a detailed description of IC-GAR model. The experimental results and discussion are in Sect. 4 and the conclusion is in Sect. 5.

2 Related works

Recommender systems have evolved over range with two main branches emerging. That general recommendation and sequential recommendation. General recommender systems do not consider the temporal nature of user interest, while sequential recommender systems are built with the dynamic nature of user interest in mind. General recommender systems can be categorized into collaborative filtering, content based and hybrid methods [16, 17]. Collaborative filtering generate recommendation for users by exploring the preferences of other related users. Content-based methods generate recommendations for users by exploring similarity between items previously consumed by the users. Hybrid methods on the other hand combines the benefit of both collaborative filtering and content-based methods. Recently, fuzzy tools have been developed for improving general recommendation [18].

Session-based recommendation is a special type of sequential recommendation where user information is not available and sessions are short. This section will present some related literature that are most relevant to our work. Related works on session-based recommendation are presented in Sect. 2.1, while graph-based recommendation systems are discussed in Sect. 2.2.

2.1 Session-based recommendation

Session-based recommendation is a sub-task of recommender systems where given the historical sequential interactions, the next item is predicted. Session-based recommendation additionally assumes that sessions cannot be tied to a particular user (anonymous user sessions). Traditional collaborative filtering models cannot be used for session-based recommendation because they do not consider the sequence of interactions. Hence, MC-based models have been extensively used [3, 5, 19, 20]. These models predict the next action in a sequence using the last action (or last few actions) and assume independence relationship between actions in a sequence. Zimdars et al. [19] used MC to extracted sequential pattern for session-based recommendation. Shani et al. [3] improved the maximum likelihood of MC transition graphs by using heuristic methods for sequential recommendation. FPMC [5] generalized MC and MF to learn sequential patterns and long-term user preference. However, MC-based models suffer from the independent assumption and an unmanageable state space when considering long sequences.

RNN solved the limitation of MC-based models. RNN can efficiently learn longer sequences and have recently shown superior performance in tasks such as machine translation [21, 22], image captioning [23, 24] and conversation systems [25]. RNN have also shown superior performance in sequential recommendation tasks such as next location [26, 27], next click [28–30] and next basket [31, 32] recommendation. Hidasi et al. [6] is the first to propose using RNN for session-based recommendation, which uses parallel mini-batches with pair-wise ranking. Tan et al. [33] improved the model by using data augmentation, privileged information and point-wise ranking loss. These models and others [34–36] only consider the last hidden state (local user preference) of the RNN for recommendation. To improve the capability of modeling a user's dynamic interest, NARM was proposed to learn both of the global and the local user preference [10]. Other models [37–39] have leveraged these two preferences and have achieved improved performance. STAMP [37] uses memory network for local and global user preference with a trilinear decomposition. LSAMN [40] proposed using hierarchical attention to balance between global user interest and sequential behavior. HLN [39] introduces a hierarchical leap network to skip preference un-related items. In addition to global and local user preference, CSRM [9] uses a memory network to incorporate neighborhood sessions.

2.2 Graph neural network based recommendation systems

Graph neural networks GNN are deep learning methods on graph structured data [41]. They learn powerful representation by using message passing technique between the nodes [15]. The main technique of GNN is to iteratively aggregate the features from the neighboring nodes with the features of the current node for a powerful node representation. GNN have achieved great success in tasks such as node classification [42–44], protein structure [45, 46] and physical systems [47, 48]. Naturally, recommendation task can be represented as a bipartite graph of user-item interactions. Several GNN models have been proposed on bipartite graphs [49–53]. Berg et al. [49] used graph auto-encoder to learn the node embeddings on a user-item bipartite graph. Ying et al. [50] improved the scalability of GNN in recommender systems by using random walks for feature aggregation. IG-MC [52] constructs one-hop subgraph based on user-item pairs to learn an inductive matrix completion method. Other models focus on both the user-item bipartite graph with additional side information graphs, such as social networks, [54–56] and the knowledge graphs [57–59]. Wu et al. [55] captured the heterogeneous information from the social and user-item graphs to model the social influence in recommendation. KGAT [59] proposed to learn the relationships in a higher-order collaborative knowledge graph. Recently, GNN have been applied on sequence data for recommendation [60–64]. SR-GNN [60] employs a gated graph neural network and an attention mechanism with bilinear decoder for recommendation. A-PGNN [62] proposed a personalized recommendation model to capture the complex item transition in a user-specific fashion. DHCN [65] replaces the directed graph used in SR-GNN with hypergraph and proposed a self-supervised learning for improved performance. GCE-GNN [66] uses epsilon neighbor and augment the long-term user preference in SR-GNN, while neglecting the short-term user preference. GAG [67] considers dynamic sessions against the static sessions and proposed using GNN with Wassertein reservoir for streaming session-base recommendation.

In this paper, our proposed model differs with the existing models from the following three points: (1) We proposed to augment an RNN-based session-based model with item co-occurrence graph which has not been considered. (2) Different from the existing GNN session-based recommendation models that construct two (incoming and outgoing) graphs for each session, we construct one undirected graph for all sessions in the training sequences. (3) We consider three sources of information i.e., global preference, local preference and item co-occurrence pattern for recommendation.

3 IC-GAR model

In this section, we present a detailed description of the proposed IC-GAR model. First, we present an overview of the model in Sect. 3.1. We then present the details of each of the three modules of IC-GAR: Encoder Module, Session Co-occurrence Module and Prediction Module in Sects. 3.2, 3.3 and 3.4, respectively. For simplicity, Table 1, shows the meaning of symbols used in the paper.

3.1 Overview of IC-GAR

Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of all items and $s = [v_{s,1}, v_{s,2}, \ldots, v_{s,m-1}]$ be the sequence list of items clicked in session s such that $v_{s,i} \in V$. Session-based recommendation aims to predict the next item that will be clicked in session s, $v_{s,m}$. The output of IC-GAR are the probability scores $\hat{\boldsymbol{y}}$ for all the candidate items, where the top-k items based on the highest probabilities are recommended as the potential next click.

The IC-GAR model is composed of three modules, Encoder, Session Co-occurrence and Prediction Modules as shown in Fig. 2. The Encoder Module learns both the local and the global preference. To model the local preference, we use the last hidden state of GRU. For the global preference, we use an attention mechanism on all hidden states of GRU to selectively model the global preference. The important contribution of IC-GAR model is the session co-occurrence representation. To model the session co-occurrence, we first construct a weighted undirected graph of the transition on all items in the training set. Each weighted edge of the graph represents the frequency of transition from one item to another. The size of the graph is of the order V^2 and for a large item size, it will slow the

Table 1 Meaning of symbols

Symbol	Meaning
V	Set of all items
$\mathbf{d}$	Hidden dimension size
$\mathbf{s}_l$	Local user preference
$\mathbf{s}_g$	Global user preference
$\mathbf{s}_s$	Item co-occurrence pattern
$\mathbf{s}_f$	Final session embedding
M_l	GCN layer update
$\mathbf{m}^*$	Final item embedding in the co-occurrence graph
$\hat{\mathbf{z}}$	Un-normalized candidate item score
$\hat{y}$	Probability scores for each item

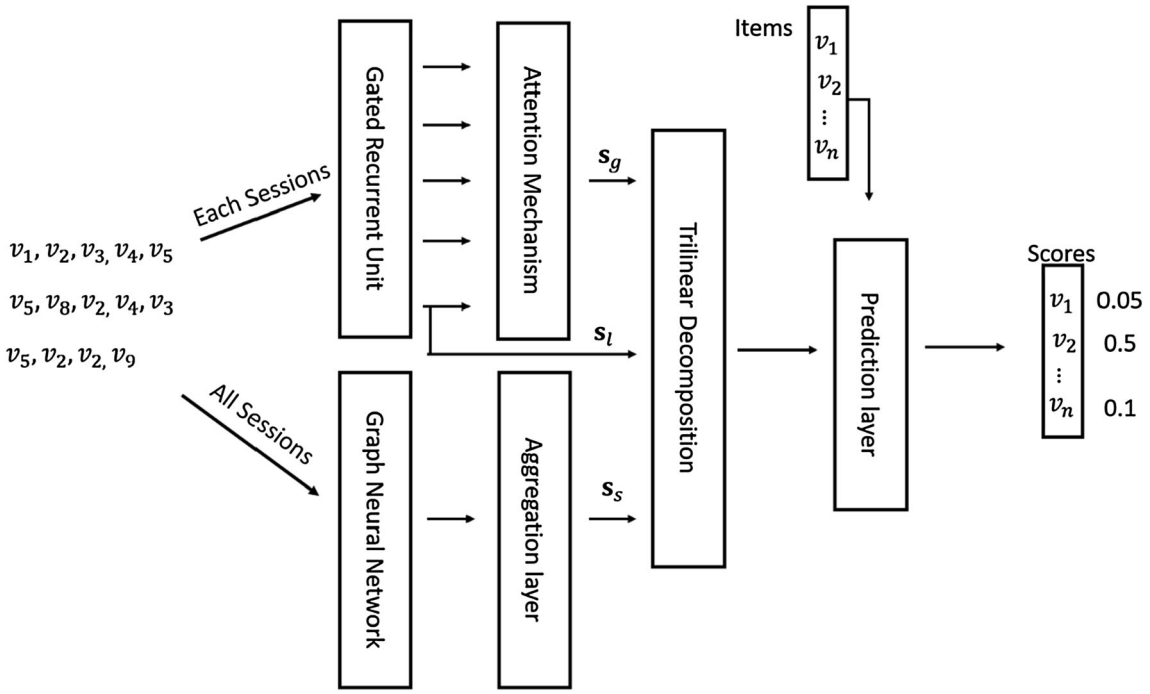

Fig. 2 Overall schematic architecture of IC-GAR model. For each session the encoder module learns the global and the local preference, $\mathbf{s}_g$ and $\mathbf{s}_l$. The session co-occurrence module first learns representations by using GCN. For the current session, the corresponding embeddings are aggregated to model session co-occurrence representation, $\mathbf{s}_s$. The prediction module decomposes the three representations and predict the ranking probabilities

computation. By using a variant of GCN, we model a lower dimension representation of the item co-occurrence graph of order Vd where $d \ll V$. The learned lower dimension representation incorporates the higher-order transition patterns between the items. By using a permutation invariant aggregation function, we form session co-occurrence representation with the representation of all the items in the current session. The session co-occurrence learns nonlinear transition patterns between the current session and all the items in the training set. Note that, we only use the sessions in the training set to construct the graph. The prediction module makes inference by first efficiently learning the final session representation using a trilinear decomposition. Finally, the probability of each next item is computed by multiplying the final session representation with candidate item and taking the softmax operation. Detailed description of each component of the model is presented next.

3.2 Encoder module

The encoder module learns both the global and the local session preference. The local preference represents the current interest of the session, while the global preference represents the changes in interest over the current session. To learn these preferences, we use a GRU which has been shown in [33] to outperform LSTM [68] and the vanilla RNN in session-based recommendation. GRU eliminates the vanishing gradient problem of the vanilla RNN by using the reset gates and the update gates. The last hidden state $\mathbf{h}_t$ of GRU is a linear combination between the previous hidden state $\mathbf{h}_{t-1}$ and a candidate state $\widehat{\mathbf{h}}_t$. It is given by:

$$\mathbf{h}_t = (1 - z_t)\mathbf{h}_{t-1} + z_t\hat{\mathbf{h}}_t, \tag{1}$$

where z_t is the update gate and is computed as:

$$z_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1}), \tag{2}$$

while $\mathbf{x}_t$ is the input at timestamp t. The candidate state can be computed as:

$$\hat{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(r_t \odot \mathbf{h}_{t-1})), \tag{3}$$

where r_t is the reset gate and is computed as:

$$r_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1}). \tag{4}$$

$\mathbf{W}_z, \mathbf{U}_z, \mathbf{W}_r, \mathbf{U}_r, \mathbf{W}$, and $\mathbf{U}$ are weight matrices of the update gate, reset gate and candidate state, respectively. The final hidden state $\mathbf{h}_n$ of the GRU represents the current interest of the session. Hence, we represent the local preference, $\mathbf{s}_l$ as:

$$\mathbf{s}_l = \mathbf{h}_n. \tag{5}$$

The global preference aims to capture the changes in interest over the current session. However, some item

clicks in the session may not truly represent a user interest, or may not contribute to current user interest. To effectively model user dynamic interest over the session, we use an attention mechanism conditioned on the last clicked item. The global preference of each session, $\mathbf{s}_g$ is thus computed by:

$$\mathbf{s}_g = \sum_{i=1}^{n} \alpha_i \mathbf{h}_i. \tag{6}$$

$\mathbf{h}_i$ is the hidden state of the GRU at timestamp i and α_i is the attention weight at i, computed as:

$$\alpha_i = \mathbf{q}^{\mathrm{T}} \sigma(\mathbf{W}_1 \mathbf{h}_t + \mathbf{W}_2 \mathbf{h}_i + b). \tag{7}$$

$\mathbf{W}_1, \mathbf{W}_2$ and $\mathbf{q}$ are learnable parameters that control the attention weights. σ is a nonlinear activation function defined as $\sigma = 1/(1 + \exp(-x))$. The encoder module converts the current session into two representations $\mathbf{s}_l$ and $\mathbf{s}_g$, the local session preference and the global, respectively.

3.3 Session co-occurrence module

The session co-occurrence module learns the transition patterns between each item in the current session and all the other items in the training sessions. The session co-occurrence improves the performance of recommendation through injecting similarity in transition patterns. The session co-occurrence module is composed of three stages, (1) Item co-occurrence graph construction. (2) Learning lower dimension representation. (3) Aggregation of the learned item co-occurrence embedding. We discuss each stage as follows:

3.3.1 Item co-occurrence graph construction

We construct a weighted undirected graph to represent the item co-occurrence patterns. The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed where $v \in \mathcal{V}$. A weighted undirected edge $(v_{i-1}, v_i) \in \mathcal{E}$ exists if v_{i-1} is clicked before or after v_i. The weights indicate the frequency of transitions between each pair of items in the training set. A weighted undirected adjacency matric $\mathbf{A}$ is then obtained for the graph. We embed each item $v \in \mathcal{V}$ into a unified embedding space, and then use a GCN to learn the higher-order transitions between the items.

3.3.2 Higher-order transition representation learning

GCN is an implementation of graph neural network based on message passing technique. The GCN model proposed in [15] updates the representation at each layer by message construction and aggregation. The update at layer l is given by:

$$\mathbf{M}_l = \mathrm{ReLU}\left(\tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{M}_{l-1} \mathbf{W}_{l-1}\right) \tag{8}$$

$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}$ is the diagonal matrix of $\tilde{\mathbf{A}}$, $\mathbf{W}_{l-1}$ is the weight matrix at layer $l-1$ and $\mathbf{M}_{l-1}$ is the representation at layer $l-1$. $\mathbf{M}_0$ is given by:

$$\mathbf{M}_0 = \mathbf{X} \tag{9}$$

$\mathbf{X}$ is the initial embedding of the items. In our case it is given by $\mathbf{V}$.

However, the GCN model in [15] was proposed for node classification task. To make the model suitable for our task, we make the following modifications. First, our update at layer l is given by:

$$\mathbf{M}_l = \mathrm{LeakyReLU}\left(\tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}} \mathbf{M}_{l-1} \mathbf{W}_{l-1}\right) \tag{10}$$

We found out that these modifications can improve the stability of the model. Secondly, similar to [51], the final embedding is given by concatenating the output of each layer. Although, [43] proposed other alternatives such as max pooling or sum pooling, concatenation can outperform these methods. The final embedding of each item in the co-occurrence graph $\mathbf{m}^*$ is thus given by

$$\mathbf{m}^* = \mathbf{m}^0 \| \ldots \| \mathbf{m}^l \tag{11}$$

where $\|$ is a concatenation operation. It controls the range of the propagation and enriches the final embedding. We also employ node and message dropout in the propagation layers to improve the robustness. The node dropout acts by dropping the nodes with p_1 probability, while the message dropout acts by dropping connections between the nodes with probability p_2.

3.3.3 Aggregation

To obtain the session co-occurrence representation, we aggregate the individual item embedding of each session. Assuming a session is given by, $s = v_1, v_2, v_3$, we obtain the session co-occurrence representation as:

$$\mathbf{s}_s = \sum_{i=1}^{3} \mathbf{m}_i^* \tag{12}$$

where $\mathbf{m}_i^*$ is the final representation of item v_i in GCN and $v_i \in s$.

3.4 Prediction module

The final prediction of the model consists of two stages. Firstly, obtaining the final session representation from the local preference, the global preference and the session co-

occurrence; secondly, obtaining the probabilities of all the candidate items for recommendation. To efficiently obtain the final session representation, we employ a trilinear decomposition given by: $< a, b, c > = a \odot (b \oplus c)$. Specifically, the final session embedding $\mathbf{s}_f$ is given by:

$$\mathbf{s}_f = \mathbf{s}_l \odot \left(\mathbf{s}_g \oplus \mathbf{s}_s\right) \tag{13}$$

where $\odot$ denotes the Hadamard product and $\oplus$ denotes the element-wise addition. The two representations $\mathbf{s}_g$ and $\mathbf{s}_s$ are conditioned on $\mathbf{s}_l$ to amplify the current user interest for recommendation.

With the embedding of each session $\mathbf{s}_f$ obtained, the candidate item $\widehat{\mathbf{z}}$ can be computed as:

$$\hat{\mathbf{z}} = \mathbf{s}_f^T \mathbf{v} \tag{14}$$

$\mathbf{v}$ is the initial embedding of all the candidate items. Softmax function is then applied to obtain the output probabilities $\widehat{\mathbf{y}}$ of the candidate items

$$\hat{\mathbf{y}} = \text{softmax}(\hat{\mathbf{z}}) \tag{15}$$

For each session, we use cross-entropy loss function between the predicted click and the ground truth. The cross-entropy loss function is defined as:

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{n} \mathbf{y_i} \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i) \tag{16}$$

where $\mathbf{y}$ is the ground truth represented by one-hot encoding. We use Back-Propagation Through Time (BPTT) to train IC-GAR model. Similar to [10, 33], we truncate the back-propagation at 19 timestamps.

4 Experimental results and performance analysis

In this section, we first describe the datasets, the state-of-the-art baseline models and the evaluation metrics for performance evaluation. We then intend to answer the following research questions.

RQ1 Does the proposed IC-GAR model achieve the state-of-the-art performance?
RQ2 What is the effect of the item co-occurrence graph on the performance of IC-GAR?
RQ3 How well does IC-GAR perform with different embedding size, the aggregation methods and the graph type?

4.1 Dataset

To evaluate the performance of IC-GAR model, we used two popular transactional datasets, namely; RetailRocket[1] and Yoochoose.[2] RetailRocket dataset contains 6 months personalized transactions from an e-commerce site. Yoochoose was published in RecSys Challenge 2015. It consists of click streams from an e-commerce site. Similar to [10, 33, 37, 60], we use the most recent 1/64 and 1/4 fractions of the Yoochoose dataset in our evaluations.

In order to filter noisy data, we filter out sessions with less than 2 items and items appearing less than 5 times in both datasets. After filtering, 37,484 items with 7,966,257 sessions remained in the Yoochoose dataset, while the RetailRocket dataset contains 46,874 items with 710,856 clicks. The summary of the dataset is given in Table 2. Following [6, 9, 69] we set the data of the last day as the test data and the remaining data for training on the Yoochoose 1/64 and Yoochoose 1/4 fractions. For RetailRocket dataset, we set the data of the last week as the test data similar to [61] and the remaining dataset for training.

4.2 Evaluation metrics

We used two accuracy metrics to evaluate the performance of all the models. Precision ($P@k$) and Mean Reciprocal Ranking ($MRR@k$) similar to previous [9, 10, 37, 60]. Both metrices evaluate the accuracy of the recommended top-k list. MRR@k additionally penalizes the ranking order of the recommended list.

***P@k*:** Mathematically, $P@k$ can be defined as:

$$P@k = \frac{n_{hit}}{N}, \tag{17}$$

where n_{hit} is the number of correctly recommended items within the top-k positions, and N is the total number of items in the test set. It measures the proportion of the test items that are correctly recommended in top-k positions within the ranking list.

MRR@k: $MRR@k$ can be defined as:

$$\text{MRR}@k = \frac{1}{N} \sum_{t \in T} \frac{1}{\text{Rank}(t)}, \tag{18}$$

where t is an item within the ranking list T. The MRR@k is set to zero if the rank of t is above k. It is the average of the reciprocal ranking of correctly recommended items. It is a better metric to evaluate the accuracy of recommender systems since the aim is to put the most relevant items at the top of the recommended list. We evaluate $P@k$ and MRR@k where $k = 5, 10$ since users are more likely to select items that appear in the top of the recommended list compared to the items with lower rankings [70, 71].

[1] https://www.kaggle.com/retailrocket/ecommerce-dataset.
[2] http://2015.recsyschallenge.com/challege.html.

Table 2 Statistics of datasets used for evaluation

Datasets	Training sessions	Test sessions	Items	Average length
RetailRocket	433,648	15,132	46,874	5.43
Yoochose 1/4	1,991,562	15,324	37,484	5.71
Yoochose 1/64	124,472	15,324	37,484	6.16

4.3 Baselines

We evaluate the performance of IC-GAR model with the following representative state-of-the-art baselines and closely related works. We use the hyperparameters in the initial paper for models with similar dataset and tune the hyperparameter for the other datasets.

BPR-MF [72] cannot be directed used for session-based recommendation because it does not consider the sequence of interactions. To use MF for session-based recommendation, latent representation of items within a session can be used to represent the session.

FMPC[3] [5] is an MC-based model for sequential recommendation. It is a state-of-the-art method for next-basket recommendation.

GRU4Rec[4] [6] first introduces RNN for session-based recommendation. It uses a GRU with pair-wise ranking and parallel mini-batches to speed-up the performance for recommendation.

NARM[5] [10] is an encoder-decoder model for session-based recommendation. It uses a GRU to learn both the user's local and global preference within each session for recommendation.

STAMP[6] [37] is an attention memory priority model that uses MLP to capture the long term and the short term user interest within the current session for session-based recommendation.

SR-GNN[7] [60] uses a GNN to inject the higher-order transition between the items in each session and learns the global and the local preference for session-based recommendation.

CSRM[8] [9] uses an inner and outer memory network for session-based recommendation. The inner memory network learns a user's interest from current session and the outer memory network uses a similarity function to learn a user's interest from the neighboring sessions.

GCE-GNN[9] [66] uses epsilon neighbor and augment the long-term user preference in SR-GNN while neglecting the short-term user preference.

[3] https://github.com/khesui/FPMC.

[4] https://github.com/hidasib/GRU4Rec.

[5] https://github.com/lijingsdu/sessionRec_NARM..

[6] https://github.com/uestcnlp/STAMP

[7] https://github.com/CRIPAC-DIG/SR-GNN.

[8] https://github.com/wmeirui/CSRM_SIGIR2019

4.4 Parameter settings

All the weight matrices and the embeddings were initialized using a Gaussian distribution with 0 mean and 0.1 standard deviation. Zero initialization was then used for all the biases. A mini-batch of size 512 was used and the epoch is set to 10. Grid search was used on all the datasets for hyperparameter selections based on MRR@10 score on the validation set. Hyperparameters in the grid search includes: learning rate η in {0.01, 0.05, 0.001, 0.005, 0.0001}, learning rate decay λ in {0.1, 0.3, 0.5, 0.7}, embedding dimension d in {50, 100, 150, 200}. Based on the average performance, we used the following hyperparameter settings in the test data: {$\eta = 0.001$, $\lambda = 0.1$, $d = 100$}. We set the number of GNN layers to 2 with the message dropout in each layer set to 20%. The node dropout is set to 40% to overcome overfitting. IC-GAR was implemented using Tensorflow.[10] Our implementation will be made available for reproducibility.[11]

4.5 Performance comparison

To evaluate the performance of the proposed IC-GAR model, we start with comparing the performance against the state-of-the-art models. We further compare the training time of the proposed model with other RNN-based state-of-the-art models.

4.5.1 Overall performance

Table 3 shows the of performance comparison with best performance shown in bold face. The following observations can be made:

- BPR-MF shows the worst performance on all the three datasets. It shows that the traditional MF methods are not sufficient for modeling user dynamic preference. FMPC is a first-order MC sequential model that only considers the last item for recommendation. FMPC outperforms BR-MF on all the three datasets, demonstrating the necessity of modeling user sequential pattern for performance enhancement.

[9] https://github.com/CCIIPLab/GCE-GNN.

[10] https://www.tensorflow.org.

[11] https://github.com/Taj-Gwadabe/IC-GAR.

Table 3 Overall performance comparison with the state-of-the-art models (values are in percentages)

	Yoochoose 1/64				Yoochoose 1/4				RetailRocket			
	MRR		Precision		MRR		Precision		MRR		Precision	
	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10
BPR-MF	17.65	18.89	24.16	31.30	15.37	16.24	21.77	22.93	13.56	14.07	21.06	27.19
FMPC	19.28	20.05	28.91	37.44	17.33	18.54	27.09	36.12	10.13	11.52	17.32	23.19
GRU4Rec	22.82	24.53	39.67	52.42	23.29	26.05	42.38	55.49	12.86	14.89	21.96	28.69
NARM	25.73	27.42	45.21	57.83	26.77	28.51	45.09	57.98	22.09	23.23	31.97	38.65
STAMP	23.55	25.17	40.09	52.96	25.26	28.32	43.16	57.67	22.41	23.34	32.87	39.72
SR-GNN	27.26	28.92	45.39	58.01	29.34	31.08	48.15	61.06	24.22	25.25	35.02	42.68
CSRM	27.13	28.24	45.82	57.96	28.64	29.14	48.09	60.54	31.92	33.05	44.44	52.70
GCE-GNN	27.27	28.98	47.58	**58.03**	29.82	31.10	48.28	61.09	32.06	33.13	45.27	53.25
IC-GAR	**32.13**	**33.38**	**48.55**	57.84	**32.74**	**34.09**	**50.17**	**60.16**	**38.67**	**39.52**	**50.26**	**56.44**

Bold values indicate the best performance

- GRU4Rec is an RNN-based model that is able to model longer sequence for recommendation. It outperforms both FMPC and BPR-MF on all the datasets demonstrating the necessity of longer sequence modeling. However, it only uses the last hidden state of the GRU for recommendation.
- NARM and STAMP both outperformed GRU4Rec on all the three datasets. It demonstrates the necessity of learning both a user's local preference and global preference for recommendation. Particularly, STAMP slightly outperforms NARM on RetailRocket dataset. On both Yoochoose datasets, NARM outperforms STAMP. This might be the result of the nature of the dataset. It also shows that RNN-based models are sufficient for session-based recommendation in most settings.
- CSRM is an RNN-based model that performed better than NARM and STMAP on all the three datasets. In addition to the local and the global preference, CSRM utilizes the neighboring sessions for improved recommendation
- SR-GNN and GCE-GNN are GNN-based models that also performed better than NARM and STAMP on all the three datasets. In addition to the local and global preference, SR-GNN utilizes the transition interaction between the items in the same session to improve the performance of the recommendation. GCE-GNN on the other hand, utilizes item level information from epsilon neighbors to augment the global preference. GCE-GNN does not consider the local preference as in the other models. Compared with CSRM, the two GNN-based models (SR-GNN and GCE-GNN) performed better on the Yoochoose 1/64 and Yoochoose 1/4 datasets. However, on the RetailRocket dataset, CSRM and GCE-GNN outperformed SR-GNN. It showed the significance of utilizing the additional information from neighboring sessions in session-based recommendation.
- IC-GAR significantly outperforms all the baseline models on MRR@5, 10 and P@5. In particular, on Yoochoose 1/64 dataset, IC-GAR outperforms the best baseline by 17.9, 15.4 and 5.9% on MRR@5, MRR@10 and P@5, respectively. On Yoochoose 1/4, IC-GAR performs better than the best baseline by 11.6, 9.7 and 4.2% on MRR@5, MRR@10 and P@5, respectively. IC-GAR outperforms all the baselines on the RetailRocket. It outperforms the best baseline by 21.1, 19.6, 13.1 and 7.1% on MRR@5, MRR@10, P@5 and P@10, respectively. Of particular importance is the performance of IC-GAR in terms on MRR@5,10. It outperforms the best baseline by 9.7–21.1% on all datasets. It clearly shows that considering the item co-occurrence patterns can significantly improve the ordering of the recommended list. IC-GAR slightly performs worse than the best baseline on both 1/64 and 1/4 Yoochoose datasets on P@10. It may be due to the fact that IC-GAR model only constructs one graph for all sessions and some local patterns may not be fully exploited with only one graph. However, GNN models are slow in training, especially when the size and number of graphs are large. The training time is reduced as only one graph is used for all the sessions.

4.5.2 Performance w.r.t to session length

The performance of session-based recommendation models may differ as the length of sessions increases or decreases. We compare the performance of IC-GAR on different session lengths. Particularly, we compare the performance

Table 4 Performance w.r.t to session length

		Metrics (%)	SR-GNN	CSRM	GCE-GNN	IC-GAR
Yoochoose 1/64	Short	P@10	61.71	55.82	61.74	59.96
		MRR@10	31.96	28.54	31.92	35.21
	Long	P@10	44.98	40.36	50.53	42.69
		MRR@10	22.16	17.13	23.19	23.31
RetailRocket	Short	P@10	56.50	57.82	57.78	58.26
		MRR@10	35.79	36.83	36.85	40.32
	Long	P@10	11.30	19.72	21.69	20.00
		MRR@10	5.47	10.00	10.91	14.95

of SR-GNN, CSRM, GCE-GNN and IC-GAR for short session and long sessions on the Yoochoose 1/64 and RetailRocket datasets with P@10 and MRR@10. Similar to SR-GNN, we divide sessions into "short" and "long" based on the average length of the session. On both datasets, we use sessions with length greater than 5 as "long" session, while the rest is used as "short" sessions. Table 4 shows the performance of SR-GNN, CSRM, GCE-GNN and IC-GAR on Yoochoose 1/64 and RetailRocket datasets for "short" and "long" session. It can be seen that across all models, the performance significantly drops for "long" session. GCE-GNN significantly outperformed other models on the Yoochoose 1/64 dataset on P@10 metrics. The performance may be attributed to the epsilon neighborhood that GCE-GNN considers. The performance of SR-GNN is of particular importance on the RetailRocket dataset for "long" session. It can be seen there is massive drop in performance which can be attributed to the maximum length on RetailRocket dataset. The performance shows that SR-GNN may not be a suitable model as the session length drastically increases. However, for "short" sessions, there is improvement in performance across all models. It shows that session-based recommendation models were designed with short sessions in mind. It also shows that, as the session length increases, there is need to consider other factors for improving performance.

4.5.3 Performance w.r.t to training time

We compare the training time of IC-GAR with the baseline models in terms of performance, namely: SR-GNN, GCE-GNN and CSRM. The training time comparison is motivated by the slow nature of training GNN models as the size and number of the graphs increases. Figure 3 shows the average training time per epoch on all the three datasets on the same GPU server. It can be seen that SR-GNN and GCE-GNN take on average twice the time required to train CSRM per epoch. The time required will significantly increase as the length of the session increases in SR-GNN and GCE-GNN due to the size of the resulting outgoing and incoming adjacency matrices that the models construct

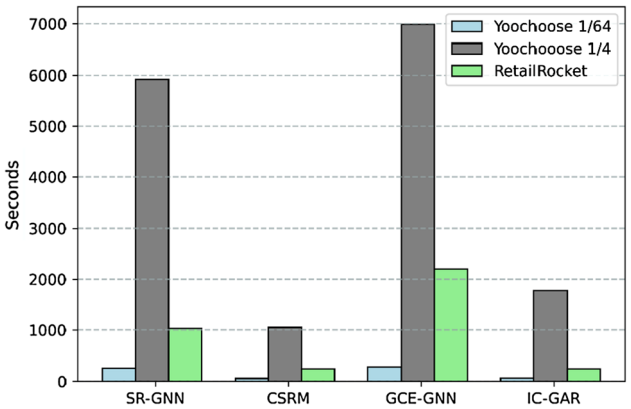

Fig. 3 Training time per epoch (best viewed in colour)

for each session. On average, IC-GAR takes less training time per epoch than CSRM despite using GNN. It can be attributed to the fact that IC-GAR only constructs one graph for the whole dataset and that the graph constructed in IC-GAR does not depend on the length of the session rather the number of items in the catalog.

4.6 Effect of item co-occurrence graph

IC-GAR distinguishes itself from other RNN-based models for session-based recommendation by constructing item co-occurrence graph using GNN. Here, we investigate the relevance of the item co-occurrence graph for session-based recommendation. Table 5 shows the performance of IC-GAR with and without the item-occurrence graph. We name the model without the item co-occurrence as SRB, while the model with item co-occurrence remains as IC-GAR. It can be seen that on all three datasets, using item co-occurrence graph significantly improves the performance. On average, there is an improvement of at least 15.7, 8.5 and 36% on Yoochoose 1/64, Yoochoose 1/4 and RetailRocket datasets, respectively. It shows that learning co-occurrence patterns can significantly improve the performance in session-based recommendation. Although, the effect of item co-occurrence graph is more significant on

Table 5 Effect of item co-occurrence graph in IC-GAR (values are in percentages)

	Metrics (%)	IC-GAR	SRB	% Improvement
Yoochoose 1/64	P@10	57.84	49.99	15.7
	MRR@10	33.38	28.69	16.3
	P@5	48.55	41.89	15.9
	MRR@5	32.13	27.60	16.4
Yoochoose 1/4	P@10	60.16	55.62	8.1
	MRR@10	34.09	31.28	9.0
	P@5	50.17	46.25	8.5
	MRR@5	32.74	30.01	9.1
RetailRocket	P@10	56.44	40.55	39.2
	MRR@10	39.52	28.97	36.4
	P@5	50.26	36.61	37.3
	MRR@5	38.67	28.44	36.0

RetailRocket, it significantly improves the performance on all datasets.

4.7 Ablation study

As various components play different roles in the performance of IC-GAR, we investigate the relevance of the different choices in the architecture. First, we study the effect of the embedding size of the GRU and GCN. We then study the effect of different aggregation methods. Finally, we study the effect of the graph type used in the GCN.

4.7.1 Effect of embedding size

For fair comparison, we used the same embedding size as the other baseline models in Table 2 for the overall performance (embedding size = 100). However, in this section we show the effect of different embedding sizes on the performance of IC-GAR. Table 6 shows the performance as embedding size varies from 50 to 200 on all the three datasets. We used the same embedding size for GCN, GRU as well as all of the weights. It can be seen that on all datasets, the performance deteriorates when the embedding size is 50. However, the performance is fairly similar with the dimensions of 100, 150 and 200. It shows that once the embedding size is sufficient, the performance is insensitive for any larger embedding size. However, as the embedding size increases, the training time and the model size increase correspondingly. Hence using embedding size of 100 was an optimal selection.

4.7.2 Effect of aggregation

Different permutation invariant aggregation methods such as concatenation, max pooling and mean pooling can be used to obtain the output of GCN. Table 7 shows the effects of concatenation, max pooling and mean pooling on the performance of IC-GAR on all three datasets, respectively. It can be seen that concatenation outperforms other aggregation methods across all metrics. Concatenation may contribute to the success of IC-GAR. We further compare the performance of these aggregation methods as the number of epochs increases. We specifically compare the performance as the number of epochs increase from 1 to 10 on P@10 and MRR@10 across all datasets. Figure 4 shows that across all the datasets, concatenation outperforms both the mean pooling and the max pooling. However, performance varies at lower epochs. On Yoochoose 1/4 and RetailRocket dataset, mean pooling outperforms other methods at 1 and 2 epochs but concatenation stabilizes to a higher accuracy as the number of epochs increases (Fig. 4).

4.7.3 Effect of graph type

Previous studies on GNN-based session-based recommendation, such as [60, 61, 73], used directed graph and

Table 6 Effect of embedding size on the performance of all three datasets (values are in percentages)

	Metrics (%)	$d = 50$	$d = 100$	$d = 150$	$d = 200$
Yoochoose 1/64	P@10	56.15	57.84	57.86	57.96
	MRR@10	32.57	33.38	33.41	33.52
	P@5	47.70	48.55	48.93	48.87
	MRR@5	31.43	32.13	32.21	32.31
Yoochoose 1/4	P@10	59.49	60.16	60.06	59.93
	MRR@10	33.80	34.09	34.04	34.11
	P@5	50.05	50.17	49.95	49.93
	MRR@5	32.55	32.74	32.69	32.78
RetailRocket	P@10	54.52	56.44	56.68	56.99
	MRR@10	38.65	39.52	39.35	39.22
	P@5	49.09	50.26	50.30	50.35
	MRR@5	37.92	38.67	38.49	38.33

Table 7 Effect of different GCN aggregation methods

	Metrics (%)	Concatenation	Mean Pooling	Max Pooling
Yoochoose 1/64	P@10	57.84	57.13	57.35
	MRR@10	33.38	33.13	33.22
	P@5	48.55	48.35	48.46
	MRR@5	32.13	31.94	32.02
Yoochoose 1/4	P@10	60.16	59.88	60.04
	MRR@10	34.09	34.10	34.13
	P@5	50.17	49.80	49.86
	MRR@5	32.74	32.78	33.76
RetailRocket	P@10	56.44	55.48	55.56
	MRR@10	39.52	39.33	39.07
	P@5	50.26	49.84	49.67
	MRR@5	38.67	38.67	38.20

modeled both the incoming and outgoing adjacency matrices. However, these models apply GNN on each session. Inspired by STAMP [37] that showed the order of interaction may not be important on online transactional datasets such as Yoochoose, we used an undirected graph for IC-GAR model, which may reduce the computational complexity introduced by using both the incoming and outgoing adjacency matrices. To show the effect of such decision, Table 8 compares the performance between the undirected graph and the directed graph (having both the incoming and outgoing graphs) on IC-GAR model. Although close performance is achieved by undirected graph and directed graph, IC-GAR reduces the computational complexity and ensures a comparable training time with non-GNN based models.

5 Discussion

In this section, we will discuss our results keeping in mind the research questions we aimed to answer. The section will discuss each of the research questions.

5.1 Does the proposed IC-GAR model achieve the state-of-the-art performance?

We conducted experiments on two publicly available datasets on two accuracy metrics to determine the performance of IC-GAR against other state-of-the-art models. Table 3 shows that IC-GAR can achieve state-of-the-art performance against RNN-based models like CSRM and GNN-based models like GCE-GNN. However, as the value of k increases, the performance of IC-GAR deteriorates on Yoochoose dataset. However, on the RetailRocket dataset, across all metrics, IC-GAR outperformed the competition. The results suggests that, performance of models differ from one dataset to another and that for industrial application, the bias and nature of the dataset need to be considered before selecting any model. We also compare the performance of IC-GAR for different session length on the Yoochoose 1/64 and RetailRocket datasets. The results showed similar trend in performance as when the whole datasets were used. However, performance of SR-GNN particularly deteriorates on "long" session for RetailRocket dataset. It suggests that, SR-GNN may not be a good model as the session length drastically increases. Finally, we compare the training time for SR-GNN, CSRM, GCE-GNN and IC-GAR on the whole sessions. Results suggest that CSRM and IC-GAR have similar time complexity, while time complexity for SR-GNN and GCE-GNN more than doubles that of the other models. The overall results suggest that IC-GAR is an efficient model that can outperform other state-of-the-art on relevant datasets.

5.2 What is the effect of the item co-occurrence graph on the performance of IC-GAR?

IC-GAR comprises of local preference, global preference and item co-occurrence graph for improved performance. We compared the performance with and without the item co-occurrence and the results suggested that the item co-occurrence graph can significantly improve the performance. The results are in line with findings of CSRM where session-level collaborative information was used to improve similar baseline. However, our model uses item-level collaborative information for improved performance.

5.3 How well does IC-GAR perform with different embedding size, the aggregation methods and the graph type?

We also study the effect of some key components in IC-GAR model. The results suggest that, for a small

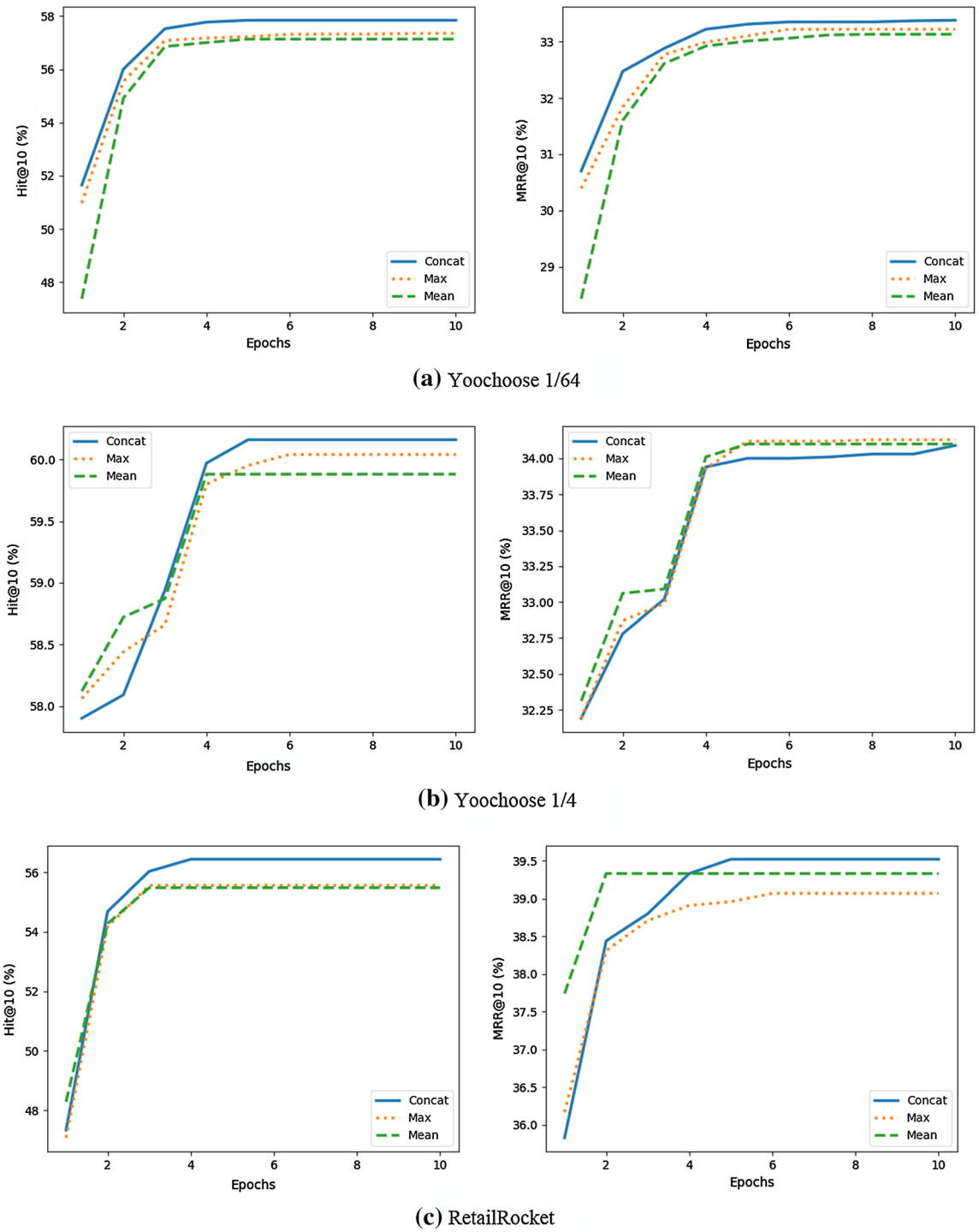

Fig. 4 Effect of GCN Aggregation Method

embedding size, IC-GAR does not reach its full performance but after reaching a sufficient embedding size, increasing it further does not significantly improve performance. Rather, as the embedding size increases, the complexity of the model further increases and slows down the training and inference. Also, the results showed that, the performance of aggregation methods vary across datasets but in our experiments, the methods (concatenation, mean-pooling and max-pooling) compared relatively have similar performances. It may be relevant to compare the performance of the aggregation methods in terms of time complexity. Finally, we compare two different graph construction methods (undirected and combined directed). The experimental results suggest that there is no significant

Table 8 Effect of using directed and undirected graph on performance of IC-GAR

	Metrics (%)	Undirected Graph	Directed graph (Incoming and Outgoing)
Yoochoose 1/64	P@10	57.84	57.69
	MRR@10	33.38	33.44
	P@5	48.55	48.72
	MRR@5	32.13	32.24
Yoochoose 1/4	P@10	60.16	60.34
	MRR@10	34.09	34.18
	P@5	50.17	50.05
	MRR@5	32.74	32.78
RetailRocket	P@10	56.44	56.62
	MRR@10	39.52	39.37
	P@5	50.26	50.19
	MRR@5	38.67	38.48

different between these methods in terms of performance. However, using both incoming and outgoing adjacency matrix can increase the computation complexity significantly.

6 Conclusion

In this paper, we proposed a novel session-based recommendation model, IC-GAR that uses a trilinear decomposition to model session representation from global preference, local preference and session co-occurrence. The session co-occurrence representation aggregates the higher-order transition patterns of all the items in the training sessions, while the global and the local preferences model user interest in the current session. Experimental results showed that IC-GAR achieved the state-of-the-art performance for session-based recommendation by using the item co-occurrence patterns.

Acknowledgements This project was partially supported by the Grant from Natural Science Foundation of China 62176247. It was also supported by the Fundamental Research Funds for the Central Universities and CAS/TWAS Presidential Fellowship for International Doctoral Students.

Declarations

Conflict of interest We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

1. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37. https://doi.org/10.1109/MC.2009.263
2. Salakhutdinov R, Mnih A (2007) Probabilistic maftrix Factorization. In: Proceedings of the 20th international conference on neural information processing systems, 2007.
3. Shani G, Heckerman D, Brafman RI (2005) An MDP-based recommender system. J Mach Learn Res 6(2005):1265–1295
4. Liang D, Zhan M, Ellis DPW (2015) Content-aware collaborative music recommendation using pre-trained neural networks. In: Proceedings of the 16th international society for music information retrieval conference, 2015.
5. Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized Markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on world wide web, 2010. https://doi.org/10.1145/1772690.1772773
6. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2016) Session-based recommendations with recurrent neural networks. In: Proceedings of the 4th international conference of learning representation, 2016.
7. Hidasi B, Quadrana M, Karatzoglou A, Tikk D (2016) Parallel recurrent neural network architectures for feature-rich session-based recommendations. In: Proceedings of the 10th ACM conference on recommender systems, 2016. https://doi.org/10.1145/2959100.2959167
8. Hidasi B, Karatzoglou A (2018) Recurrent neural networks with Top-k gains for session-based recommendations. In: Proceedings of the 27th ACM international conference on information and knowledge management, 2018. https://doi.org/10.1145/3269206.3271761
9. Wang M, Chen Z, Ren P, et al (2019) A collaborative session-based recommendation approach with parallel memory modules. In: Proceedings of the 42nd international conference on research and development in information retrieval, 2019. https://doi.org/10.1145/3331184.3331210
10. Li J, Ren P, Chen Z et al (2017) Neural attentive session-based recommendation. In: Proceedings of 26th ACM international conference on information and knowledge management, 2017. https://doi.org/10.1145/3132847.3132926
11. Liu Y, Cao X, Yu Y (2016) Are you influenced by others when rating? Improve rating prediction by conformity modeling. In: Proceedings of the 10th ACM conference on recommender systems, 2016. https://doi.org/10.1145/2959100.2959141
12. Wang T, Wang D (2014) Why Amazon's ratings might mislead you: the story of herding effects. Big Data. https://doi.org/10.1089/big.2014.0063
13. Tang J, Gao H, Liu H (2012) MTrust: discerning multi-faceted trust in a connected world. In: Proceedings of the 5th ACM

international conference on web search and data mining, 2012. https://doi.org/10.1145/2124295.2124309
14. Chaney AJB, Blei DM, Eliassi-Rad T (2015) A probabilistic model for using social networks in personalized item recommendation. In: Proceedings of the 9th ACM conference on recommender systems, 2015. https://doi.org/10.1145/2792838.2800193
15. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th international conference on learning representations, 2017.
16. Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) Recommender systems survey. Knowl-Based Syst 46:109–132. https://doi.org/10.1016/j.knosys.2013.03.012
17. Lu J, Wu D, Mao M et al (2015) Recommender system application developments: a survey. Decis Support Syst 74:12–32. https://doi.org/10.1016/j.dss.2015.03.008
18. Yera R, Martínez L (2017) Fuzzy tools in recommender systems: a survey. Int J Comput Intell Syst 10:776–803. https://doi.org/10.2991/ijcis.2017.10.1.52
19. Zimdars, Andrew; Chickering M, Christopher M (2001) Using temporal data for making recommendations. In: Proceedings of the 17th conference on uncertainty in artificial intelligence, 2001.
20. Mobasher B, Dai H, Luo T, Nakagawa M (2002) Using sequential and non-sequential patterns in predictive web usage mining tasks. In: Proceedings of IEEE international conference of data mining, 2002. https://doi.org/10.1109/ICDM.2002.1184025
21. Cho K, Van Merriënboer B, Gulcehre C, et al (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of 2014 conference on empirical methods in natural language processing, 2014.
22. Bahdanau D, Cho KH, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: Proceedings of 3rd international conference on learning representations, 2015.
23. Mao J, Xu W, Yang Y, et al (2015) Deep captioning with multimodal recurrent neural networks (m-RNN). In: Proceedings of 3rd international conference on learning representations, 2015.
24. Karpathy A, Fei-Fei L (2017) Deep visual-semantic alignments for generating image descriptions. IEEE Trans Pattern Anal Mach Intell 39(4):664–676. https://doi.org/10.1109/TPAMI.2016.2598339
25. Weng Y, Miryala SS, Khatri C et al (2020) Joint contextual modeling for ASR correction and language understanding. In: Proceedings of IEEE international conference on acoustics, speech and signal processing, 2020. https://doi.org/10.1109/ICASSP40776.2020.9053213
26. Liu Q, Wu S, Wang L, Tan T (2016) Predicting the next location: a recurrent model with spatial and temporal contexts. In: Proceedings of 30th AAAI conference on aritifical intelligence, 2016.
27. Zhao P, Zhu H, Liu Y et al (2020) Where to go next: a spatio-temporal gated network for next POI recommendation. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2020.3007194
28. Zhang Y, Dai H, Xu C, et al (2014) Sequential click prediction for sponsored search with recurrent neural networks. In: Proceedings of the 28th AAAI conference on artificial intelligence, 2014.
29. Deng W, Ling X, Qi Y, et al (2018) Ad click prediction in sequence with long short-term memory networks: An externality-aware model. In: Proceedings of the 41st International Conference on Research and Development in Information Retrieval, 2018. https://doi.org/10.1145/3209978.3210071
30. Zhang L, Wang P, Li J et al (2021) Attentive hybrid recurrent neural networks for sequential recommendation. Neural Comput Appl. https://doi.org/10.1007/s00521-020-05643-7
31. Yu F, Liu Q, Wu S, et al (2016) A dynamic recurrent model for next basket recommendation. In: Proceedings of the 39th international conference on research and development in information retrieval, 2016. https://doi.org/10.1145/2911451.2914683
32. Hu H, He X, Gao J, Zhang ZL (2020) Modeling personalized item frequency information for next-basket recommendation. In: Proceedings of the 43rd international conference on research and development in information retrieval, 2020. https://doi.org/10.1145/3397271.3401066
33. Tan YK, Xu X, Liu Y (2016) Improved recurrent neural networks for session-based recommendations. In: Proceedings of the 1st workshop on deep learning for recommender systems, 2016. https://doi.org/10.1145/2988450.2988452
34. Tuan TX, Phuong TM (2017) 3D convolutional networks for session-based recommendation with content features. In: Proceedings of the 11th ACM conference on recommender systems, 2017. https://doi.org/10.1145/3109859.3109900
35. Wu S, Ren W, Yu C, et al (2016) Personal recommendation using deep recurrent neural networks in NetEase. In: Proceedings of IEEE 32nd international conference on data engineering, 2016. https://doi.org/10.1109/ICDE.2016.7498326
36. Smirnova E, Vasile F (2017) Contextual sequence modeling for recommendation with Recurrent Neural Networks. In: Proceedings of 2nd workshop on deep learing for recommender systems, 2017. https://doi.org/10.1145/3125486.3125488
37. Liu Q, Mokhosi R, Zeng Y, Zhang H (2018) STAMP: Short-term attention/memory priority model for session-based recommendation. In: Proceedings of the 24th international conference on knowledge discovery and data mining, 2018. https://doi.org/10.1145/3219819.3219950
38. Song J, Shen H, Ou Z, et al (2019) ISLF: Interest shift and latent factors combination model for session-based recommendation. In: Proceedings of 28th international joint conference on artificial intelligence, 2019. https://doi.org/10.24963/ijcai.2019/799
39. Guo C, Zhang M, Fang J, et al (2020) Session-based recommendation with hierarchical leaping networks. In: Proceedings of the 43rd international conference on research and development in information retrieval, 2020. https://doi.org/10.1145/3397271.3401217
40. Chen D, Zhang R, Qi J, Yuan B (2019) Sequence-aware recommendation with long-term and short-term attention memory networks. In: Proceedings of the 20th IEEE international conference on mobile data management (MDM), 2019.
41. Scarselli F, Gori M, Tsoi AC et al (2009) The graph neural network model. IEEE Trans Neural Networks. https://doi.org/10.1109/TNN.2008.2005605
42. Veličković P, Casanova A, Liò P et al (2018) Graph attention networks. In: Proceedings of the 6th international conference on learning representations, 2018.
43. Xu K, Li C, Tian Y et al (2018) Representation learning on graphs with jumping knowledge networks. In: Proceedings of the 35th international conference on machine learning, 2018.
44. Ying R, Morris C, Hamilton WL et al (2018) Hierarchical graph representation learning with differentiable pooling. In: Proceedings of the 32nd international conference on neural information processing systems, 2018.
45. Fout A, Byrd J, Shariat B, Ben-Hur A (2017) Protein interface prediction using graph convolutional networks. In: Proceedings of 31st international conference on neural information processing systems, 2017.
46. Gligorijevic V, Douglas Renfrew P, Kosciolek T et al (2019) Structure-based function prediction using graph convolutional networks. bioRxiv preprint bioXiv:https://doi.org/10.1101/786236v2
47. Battaglia P, Pascanu R, Lai M et al (2016) Interaction networks for learning about objects, relations and physics. In: Proceedings

of 30th international conference on neural information processing systems, 2016.
48. Sanchez-Gonzalez A, Heess N, Springenberg JT et al (2018) Graph networks as learnable physics engines for inference and control. In: Proceedings of the 35th international conference on machine learning, 2018.
49. Berg R van den, Kipf TN, Welling M (2017) Graph convolutional matrix completion. arXiv preprint axXiv:1706.02263
50. Ying R, He R, Chen K, et al (2018) Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th international conference on knowledge discovery and data mining, 2018. https://doi.org/10.1145/3219819.3219890
51. Wang X, He X, Wang M, et al (2019) Neural graph collaborative filtering. In: Proceedings of the 42nd international conference on research and development in information retrieval, 2019. https://doi.org/10.1145/3331184.3331267
52. Zhang M, Chen Y (2019) Inductive Matrix Completion based on graph neural networks. In: Proceedings of the 8th international conference on learning representations, 2019.
53. Tao Z, Wei Y, Wang X et al (2020) MGAT: Multimodal graph attention network for recommendation. Inf Process Manage. https://doi.org/10.1016/j.ipm.2020.102277
54. Fan W, Ma Y, Li Q et al (2019) Graph neural networks for social recommendation. In: Proceedings of the world wide web conference, 2019. https://doi.org/10.1145/3308558.3313488
55. Wu L, Hong R, Sun P et al (2019) A neural influence diffusion model for social recommendation. In: Proceedings of the 42nd international conference on research and development in information retrieval, 2019. https://doi.org/10.1145/3331184.3331214
56. Wu L, Li J, Sun P et al (2020) DiffNet++: A neural influence and interest diffusion network for social recommendation. arXiv preprint arXiv:2002.0084
57. Wang H, Leskovec J, Zhang F et al (2019) Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In: Proceedings of the 25th international conference on knowledge discovery and data mining, 2019. https://doi.org/10.1145/3292500.3330836
58. Wang H, Zhao M, Xie X et al (2019) Knowledge graph convolutional networks for recommender systems. Proceedings of the World Wide Web Conference 2019. https://doi.org/10.1145/3308558.3313417
59. Wang X, He X, Cao Y, et al (2019) KGAT: Knowledge graph attention network for recommendation. In: Proceedings of the 25th international conference on knowledge discovery and data mining, 2019. https://doi.org/10.1145/3292500.3330989
60. Wu S, Tang Y, Zhu Y et al (2019) Session-based recommendation with graph neural networks. In: Proceedings of AAAI conference on artificial intelligence, 2019. https://doi.org/10.1609/aaai.v33i01.3301346
61. Xu C, Zhao P, Liu Y et al (2019) Graph contextualized self-attention network for session-based recommendation. In: Proceedings of the 28th international joint conference on artificial intelligence, 2019. https://doi.org/10.24963/ijcai.2019/547
62. Wu S, Zhang M, Jiang X et al (2019) Personalizing graph neural networks with attention mechanism for session-based recommendation. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2020.3031329
63. Pan Z, Cai F, Chen W et al (2020) Star Graph Neural Networks for Session-Based Recommendation. In: Proceedings of the 29th ACM international conference on information and knowledge management, 2020. https://doi.org/10.1145/3340531.3412014
64. Tao Y, Wang C, Yao L et al (2021) Item trend learning for sequential recommendation system using gated graph neural network. Neural Comput Appl. https://doi.org/10.1007/s00521-021-05723-2
65. Xia X, Yin H, Yu J, et al (2020) Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation. In: Proceedings of the 35th AAAI conference on artificial intelligence, 2021. http://arxiv.org/abs/2012.06852.
66. Wang Z, Wei W, Cong G et al (2020) Global context enhanced graph neural networks for session-based recommendation. In: Proceedings of the 43rd international conference on research and development in information retrieval, 2020. https://doi.org/10.1145/3397271.3401142.
67. Qiu R, Yin H, Huang Z, Chen T (2020) GAG: global attributed graph neural network for streaming session-based recommendation. In: Proceedings of the 43rd International conference on research and development in information retrieval, 2020. https://doi.org/10.1145/3397271.3401109
68. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
69. Song B, Zhang W, Cao Y, Xu C (2019) Session-based recommendation with hierarchical memory networks. In: Proceedings of the 28th ACM international conference on information and knowledge management, 2019. https://doi.org/10.1145/3357384.3358120
70. Klöckner K, Wirschum N, Jameson A (2004) Depth- and breadth-first processing of search result lists. In: Proceedings of conference on human factors in computing systems, 2004. https://doi.org/10.1145/985921.986115
71. O'Brien M, Keane MT (2006) Modeling result-list searching in the world wide web: the role of relevance topologies and trust bias. In: Proceedings of the 28th annual conference of the cognitive science society, 2006.
72. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th conference on uncertainty in artificial intelligence, 2009.
73. Qiu R, Huang Z, Li J, Yin H (2019) Rethinking the item order in session-based recommendation with graph neural networks. In: Proceedings of the 28th ACM international conference on information and knowledge management, 2019. https://doi.org/10.1145/3357384.3358010