Knowledge-Based Systems 251 (2022) 109204

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Time enhanced graph neural networks for session-based recommendation

Gu Tang [a], Xiaofei Zhu [a,*], Jiafeng Guo [b], Stefan Dietze [c,d]

[a] College of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China
[b] Institute of Computing Technology, Chinese Academy of Science, Beijing 100190, China
[c] Knowledge Technologies for the Social Sciences, Leibniz Institute for the Social Sciences, Cologne 50667, Germany
[d] Institute of Computer Science, Heinrich-Heine-University Düsseldorf, Düsseldorf 40225, Germany

ARTICLE INFO

Article history:
Received 18 November 2021
Received in revised form 31 May 2022
Accepted 1 June 2022
Available online 7 June 2022

Keywords:
Session-based recommendation
Graph neural network
Highway network
Temporal interest attention network

ABSTRACT

Session-based recommendation (SBR) is a challenging task, aiming at recommending items according to the behavior of anonymous users. Previous research efforts mainly focus on capturing sequential transitions between consecutive items via recurrent neural networks (RNN) or modeling the complex transitions between non-adjacent items based on graph neural networks (GNN). Although these works have achieved encouraging performance on solving the session-based recommendation problem, few efforts have been dedicated to exploring the rich information related to the shifts of user interests within the transition relationships, which is the research gap we attempt to bridge in this work. In this paper, we propose a novel model, named Time Enhanced Graph Neural Networks (TE-GNN), which attempts to capture the complex user interest shift patterns within sessions. In TE-GNN, we construct a Time Enhanced Session Graph (TES-Graph) where transition relationships between items are treated adaptively with respect to the degree of user interest drift. In addition, a novel Temporal Graph Convolutional Network (T-GCN) is designed to learn item embeddings based on the TES-Graph. Moreover, we also introduce a Temporal Interest Attention Network (TIAN) to model the complex transition of items with a common user interest. Extensive experiments have been conducted on four widely used benchmark datasets, i.e., Diginetica, Tmall, Nowplaying, and Retailrocket, and the results show that our proposed approach TE-GNN significantly outperforms previous state-of-the-art baseline methods. The implementation of TE-GNN is available in https://github.com/GuTang1997/TE-GNN.

1. Introduction

Recommender systems play a key role in e-commerce platforms, such as Alibaba, Tiktok and Youtube, which aim at helping users easily obtain their desired information. Early research works include Markov-chain-based methods [1,2] and collaborative filtering (CF) based methods [3,4]. The Markov-chain-based methods explore the sequential transaction data for the task of SBR. For example, FPMC [1] extends matrix factorization by incorporating a first-order Markov chain to model both sequential behavior and long-term user preference. As these Markov-chain-based methods employ a strong independent assumption, which predicts user's next item only based on the previous items, they inevitably suffer from the shifts of user interests or noisy items. The CF-based methods mainly focus on exploiting the past user-item interactions, e.g., ratings or clicks, to learn latent features for both user and item. One limitation of CF-based methods is that they mainly rely on user long-term historical interactions, which may be unavailable in real-world applications.

Later, recurrent neural networks, which have been proven effective in capturing user's general interests from previous clicks, have achieved encouraging performance on solving the session-based recommendation problem [5–9]. For example, Hidasi et al. [5] model the short-term preferences with gated recurrent units (GRUs [10]) and propose a RNN-based method GRU4Rec. Li et al. [8] explore a hybrid encoder with an attention mechanism to model user's sequential behavior and capture her main intent in the ongoing session. Liu et al. [9] propose to simultaneously capture both user's general interests from the long-term memory of a session and user's current interests from the short-term memory of the last click. Although these methods have achieved promising performance, the major limitation is that they only capture sequential transitions between consecutive actions, while cannot effectively model the complex transitions between non-adjacent actions.

Recently, graph neural networks (GNNs) [5,11–14] have been widely adopted to model the complex transition relationship between items without direct connections. For example, Wu

* Corresponding author.
E-mail addresses: gutang@2020.cqut.edu.cn (G. Tang), zxf@cqut.edu.cn (X. Zhu), guojiafeng@ict.ac.cn (J. Guo), stefan.dietze@gesis.org (S. Dietze).

https://doi.org/10.1016/j.knosys.2022.109204
0950-7051/

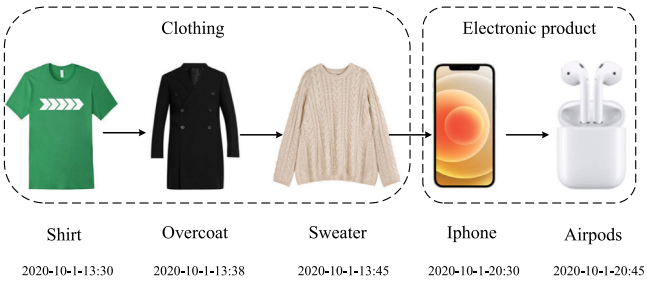

Fig. 1. An example session.

et al. [15] propose to construct a directed graph (i.e., session graph) for each session sequence, and utilize GNNs to capture the complex transitions of items and learn the embedding vectors of items. Pan et al. [16] propose a star graph neural network (SGNN), which solves the long-range information propagation issue by adding a star node to account for non-adjacent items. Wang et al. [17] exploit the transition information between items over all sessions for better capturing global contextual information. Although these session graph based methods have achieved state-of-the-art performance, they treat each transition relationship between items equally and neglect the rich user interest drift information within the transition relationships. Considering Fig. 1 as an example, where we can observe that: (1) The transition relationship between 'Sweater' and 'Iphone' is weaker than that of the transition relationship between 'Iphone' and 'Airpods' as the user interest drift of the former is considerably larger than that of the latter; (2) The user interest drift is strongly correlated with the time interval, i.e., a larger time interval between two consecutive items indicates a higher degree of user interest drift; (3) Items (e.g., 'Shirt', 'Overcoat', and 'Sweater') within a close time interval usually have a similar user interest (e.g., 'Clothing') involving a smaller interest drift.

Based on the above observations, in this paper, we propose to capture the complex pattern of user interest drift. In particular, we propose a novel Time Enhanced Session Graph (TES-Graph), which is a variant of the session graph [15] by further incorporating the degree of user interest drift into the edge of the session graph. The major difference between TES-Graph and the conventional session graph is that the former adaptively models these transition relationships while the latter treats them equally. Consequently, we design a Temporal Graph Convolutional Network (T-GCN) to learn item embeddings on the TES-Graph. In addition, to model items representing similar user interests, we introduce a Temporal Interest Attention Network (TIAN), in which items within a close time interval are separated into the same user interest bin. As items in the same bin share a common user interest, the information of each bin will be incorporated into its corresponding items to learn better item representations.

We conducted extensive experiments on four widely used real-world benchmark datasets including Diginetica, Tmall, Nowplaying and Retailrocket. Results demonstrate that our proposed approach consistently outperforms the state-of-the-art baselines with a large margin. The main contributions of this work are summarized as follows:

- We propose to model rich user interest drift information by incorporating a novel Time Enhanced Session Graph (TES-Graph), in which adaptive weights reflecting the user interest drift are estimated for each edge.
- A tailored multi-layer Temporal Graph Convolutional Network (T-GCN) based on the TES-Graph is developed to learn informative item representations.
- We also design a Temporal Interest Attention Network (TIAN) to capture the common user interest pattern of items within a session based on temporal information.
- Extensive experiments have been conducted on four widely used real-world benchmark datasets, including Diginetica, Tmall, Nowplaying and Retailrocket. The results demonstrate that our proposed approach TE-GNN is significantly superior to all state-of-the-art baseline.

2. Related work

Although collaborative filtering (CF) [18,19] is not specifically developed for the task of session-based recommendation, it has been widely used in recommendation systems by modeling user's historical global interactions with items [3,20,21]. One of the representative methods for session-based recommendation is Item-KNN [3], which relies on capturing items' global co-occurrence relationship. The main limitation of Item-KNN is that it neglects the sequential information of items within a session. Later, Rendle et al. [1] propose FPMC to capture the sequential behavior and long-term interest of users by applying Markov chains and matrix decomposition. Based on FPMC, Wang et al. [22] further propose a hierarchical representation model (HRM), which combines both user's representation and user's behavior sequence information hierarchically to improve recommendation performance.

In recent years, various deep learning based methods have been proposed and shown great performance improvement compared to previous models. For example, Hidasi et al. [5] propose GRU4REC, and apply a multi-layer Gated Recurrent Unit (GRU [10]) to model the sequential behavior of users. Li et al. [8] extend GRU4REC and propose the model NARM which combines GRU and attention mechanism to model the sequential behavior of users. Liu et al. [9] attempt to address the user interests drift issue by capturing both a user's general interests from the long-term memory of a session and her current interests from the short-term memory of the last-click. Kang et al. [23] propose a self-attention based sequential model (SASRec) to capture long-term semantics using the attention mechanism. Zhang et al. [24] propose a Gating Augmented Capsule Network (GAC) to exploit both personalized item- and factor-level transitions in a fine-grained manner. Song et al. [25] leverage a novel attention mechanism to capture the user's attention weight at each timestamp, and obtain the long-term interest representation based on a weighted sum vector of all clicked item embeddings. Wang et al. [26] consider the collaborative modeling in session-based recommendations with an end-to-end neural model. Pan et al. [27] utilize a modified self-attention mechanism to get a better long-term representation of session.

More recently, some graph neural network (GNN) based methods [15,17,28–33] have shown promising performance in the task of session-based recommendation. Wu et al. [15] propose SR-GNN which builds directed graphs from historical session sequences and generates item embeddings by leveraging GNN to model transitions of items. With the great success of SR-GNN, a variants of SR-GNN have been proposed and achieved the state-of-the-art performance. SHARE [30] considers each session as a hypergraph so as to model the item correlations defined by various contextual windows in the session simultaneously. DMGCF [29] constructs user graph, item graph and user-item bipartite graph, and combines a dual-path graph convolution network on the three graphs for collaborative filtering. SGNN [31] attempts to simulate the users' behavior patterns from a spatiotemporal perspective. SGNN is composed of two sub-modules, i.e., spatiotemporal session graph and preference-aware attention. CaSe4SR [28] builds an item graph and a category graph based on session sequence and

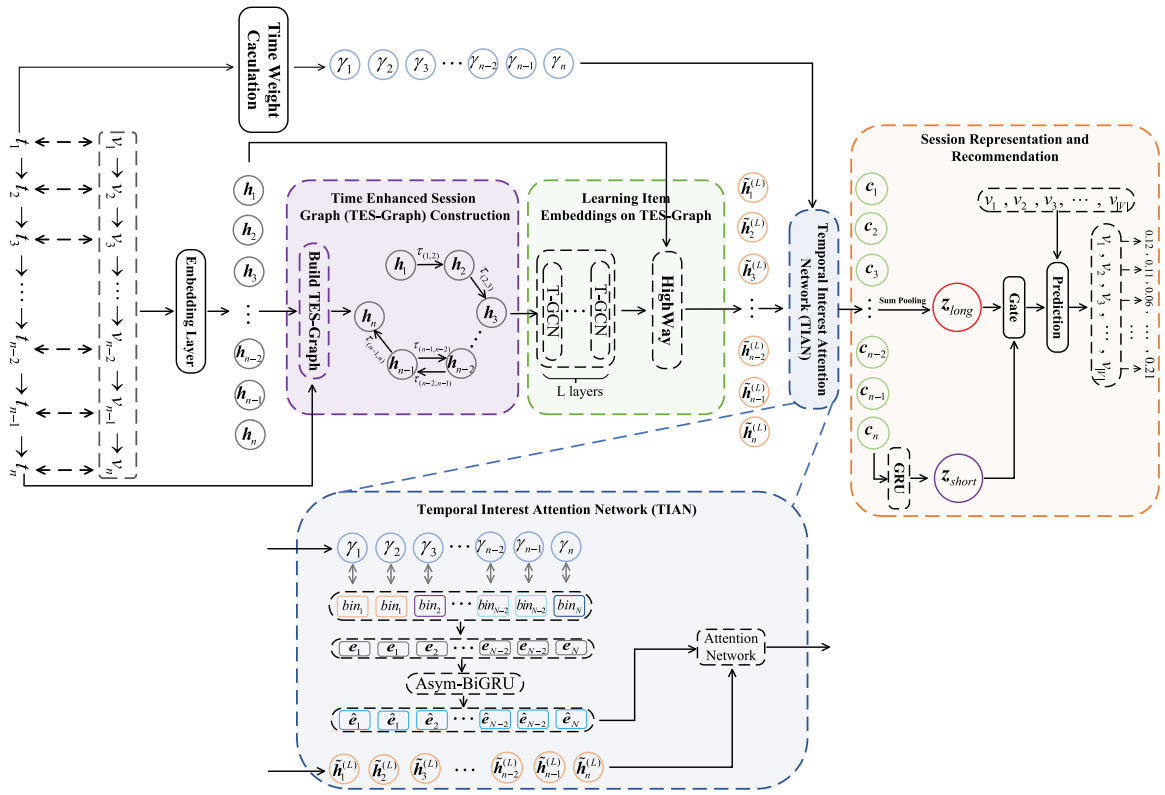

Fig. 2. The framework of our proposed model TE-GNN. It consists of four main components, including the construction of TES-Graph, learning item embeddings on TES-Graph, the temporal interest attention network, and the session representation and recommendation.

item category sequence, the category graph reduces item-level user behavior noises and makes user's interest clearer. GFE [32] develops sequential-aware interest and knowledge-aware interest modules to capture intrinsic interests and external potential interests of user. Moreover, GFE proposes a hierarchical self-attention mechanism to exploit the high-order semantic information from knowledge paths so as to discover user's dynamic preferences evolution. HDMA [33] measures the item-level similarity between users and their friends, proposes to exploit users' individual interests based on capturing the aspect difference between users' interests and friends' interests. KA-MemNN [34] develops a key-array memory network with a hierarchical intent tree to model coarse-to-fine user intents and a hierarchical semantic component to explore the multi-intent of user. TAGNN [35] jointly models user interests given a certain target item as well as complex item transitions in sessions to make recommendation. GCE-GNN [17] uses a GNN model on session graph to learn session-level item embeddings within the current session and employs a session-aware attention mechanism on global graph to learn global-level item embeddings over all sessions.

The main difference of our work with the above state-of-the-art approaches is that previous works mainly model user interests in a coarse-grained manner while neglect the user interests drift information considerably, e.g., they usually treat the transitions between two consecutive items equally. In our proposed model, we take into account the different degrees of users' interests drift between two consecutive items in a fine-grained manner by building a Time Enhanced Session Graph (TES-Graph). Then we apply a Temporal Graph Convolutional Network (T-GCN) on TES-Graph to better capture user interests within a session. In addition, we assume that items within a short time interval have similar user interest pattern, and further develop a Temporal Interest Attention Network (TIAN) to enhance item representations via capturing the similar user interest pattern of items within a session based on the temporal information.

3. Approach

In this section, we first describe the session-based recommendation task, and then introduce each module of our proposed model Time Enhanced Graph Neural Networks (TE-GNN). The framework of TE-GNN is shown in Fig. 2, which consists of four main components: (1) Time Enhanced Session Graph (TES-Graph) construction; (2) Learning item embeddings on TES-Graph; (3) Temporal Interest Attention Network (TIAN); (4) Session representation and recommendation.

3.1. Problem statement

Let $V = \{v_1, v_2, \ldots, v_{|V|}\}$ denote all unique items appearing in all sessions, where $|V|$ is the number of all unique items. $S = (v_1, v_2, \ldots, v_n)$ is an anonymous session which consists of n items, where $v_i \in V$ is the ith item in the session. The goal of session-based recommendation is to recommend top-K items that the user is most likely to click based on the current session.

3.2. Time Enhanced Session Graph (TES-Graph)

The Time Enhanced Session Graph (TES-Graph) reveals sequential patterns over transition relationships of consecutive items in the session. The main difference between the TES-Graph and the session graph [15,17] is that TES-Graph further captures rich user interest drift information by assigning each edge a weight reflecting the user interest drift. The TES-Graph is defined as follows: given a session $S = (v_1, v_2, \ldots, v_n)$, let $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, \mathcal{W}_s)$ be the corresponding TES-Graph, where $\mathcal{V}_s$ denotes the clicked items in the session S, $\mathcal{E}_s$ denotes the edge set, and $\mathcal{W}_s$ is the edge weight matrix. Each node represents an item $v_i \in \mathcal{V}_s$, and each edge $(v_{i-1}, v_i) \in \mathcal{E}_s$ denotes an adjacent relationship between two consecutive items v_{i-1} and v_i in the session, which

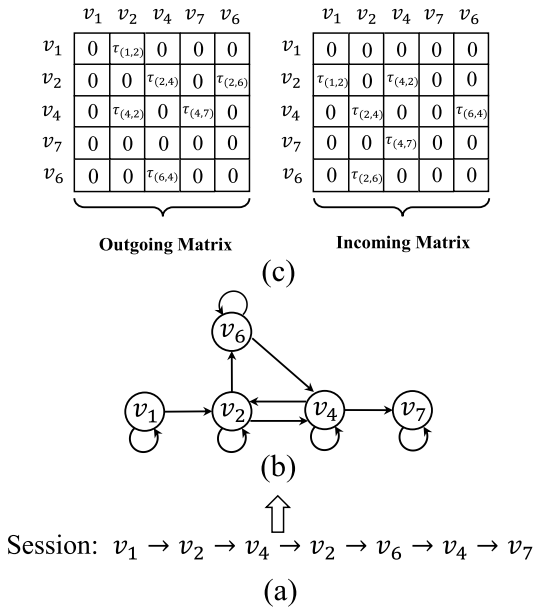

Fig. 3. The construction process of a TES-Graph for (a) an example session $S = (v_1, v_2, v_4, v_2, v_6, v_4, v_7)$, (b) its corresponding TES-Graph, as well as (c) its corresponding incoming matrix and outgoing matrix.

can be represented by an incoming matrix $\mathbf{A}^I$ and an outgoing matrix $\mathbf{A}^O$. Each edge (v_{i-1}, v_i) has a weight $\tau_{(i-1,i)} \in \mathcal{W}_s$. Similar to [17], a self-loop is added to each node, which is represented by a self-connection matrix $\mathbf{A}^S$ (i.e., an identity matrix). Fig. 3 shows the construction process of a TES-Graph for an example session $S = (v_1, v_2, v_4, v_2, v_6, v_4, v_7)$. Based on this session, we have the node set $\mathcal{V}_s = \{v_1, v_2, v_4, v_6, v_7\}$ and the edge set $\mathcal{E}_s = \{(v_1, v_2), (v_2, v_4), \ldots, (v_4, v_7)\} \cup \{(v_1, v_1), (v_2, v_2), \ldots, (v_7, v_7)\}$. Note that the edge set consists of the edges between two consecutive items as well as the self-loop of each node. The edge weight matrix $\mathcal{W}_s$ is measured based on the time interval between two consecutive items (as defined in Eq. (1)).

As pointed out in many previous research works [36], user interest drift is strongly correlated with the time interval between two consecutive items of her ongoing session, i.e., a higher time interval between two consecutive items corresponds to a higher user interest drift, and vice versa. For example, in Fig. 1, the session has 5 consecutive items (*Shirt* $\xrightarrow{8'}$ *Overcoat* $\xrightarrow{7'}$ *Sweater* $\xrightarrow{405'}$ *Iphone* $\xrightarrow{15'}$ *Airpods*), and the time interval between 'Sweater' and 'Iphone' is 405 min (high user interest drift), which is much larger than the time interval of 15 min between 'Iphone' and 'Airpods' (small user interest drift). Based on this observation, we calculate the edge weight matrix $\mathcal{W}_s$ as follows: for each session $S = (v_1, v_2, \ldots, v_n)$, we first obtain its timestamp $T = (t_1, t_2, \ldots, t_n)$, and measure the weight of each edge in the TES-Graph based on the degree of user interest drift, which is calculated based on the time interval corresponding to that edge. In our work, we assume the edge weight will decay according to its corresponding time interval between two consecutive items in a session, and we leverage Newton's law of cooling to calculate the edge weight. To be specific, according to Newton's law of cooling, the edge weight (i.e., temperature) will cool from the highest value (i.e., initial temperature) D_{init} to the lowest value (i.e., final temperature) D_{final}. In our experiments, we empirically set $D_{init} = 0.98$ and $D_{final} = 0.01$. It is noting that two items with a large time interval would also maintain common interest to some extent, therefore we set a small weight (i.e., 0.01) rather than 0 to edges even their corresponding items' time intervals are large. Besides, we also set the edge weight to 1 for the self-loop edge. Formally, for an edge between two consecutive items v_i and v_j in a session, the edge weight $\tau_{(i,j)}$ is defined as follows:

$$\tau_{(i,j)} = \begin{cases} e^{-\alpha(|t_j - t_i| + l)} & if: i \neq j, \\ 1 & if: i = j. \end{cases} \tag{1}$$

$$l = \frac{-ln(D_{init})}{\alpha}, \tag{2}$$

$$\alpha = \frac{ln(D_{init}/D_{final})}{m}, \tag{3}$$

$$m = t_n - t_1, \tag{4}$$

where D_{init} and D_{final} are two pre-specified constants indicating the initial and final edge weights which are used for the decay. α represents the decay constant, and l is the left offset which is used to make $\tau_{(i,j)}$ adaptable to the different sessions.

3.3. Temporal Graph Convolutional Network (T-GCN)

In this subsection, we introduce how to learn item embedding based on the constructed TES-Graph. We first map each item in the session $S = (v_1, v_2, \ldots, v_n)$ to an embedding sequence $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$, where $\mathbf{h}_i \in \mathbb{R}^d$ is a d-dimensional representation of item v_i. Then we apply a multi-layer Temporal Graph Convolutional Network (T-GCN) which is utilized to learn item representations on top of the TES-Graph. A single layer T-GCN will aggregate the information of the item itself and its first-order neighbors. Inspired by LightGCN [37], in T-GCN we implement a simplified GCN (i.e., $\mathbf{H}^{(l)} = \mathbf{A}\mathbf{H}^{(l-1)}$) rather than the existing GCN (i.e., $\mathbf{H}^{(l)} = \sigma(\mathbf{A}\mathbf{H}^{(l-1)}\mathbf{W}^{(l-1)})$) by discarding the two most common used mechanisms (i.e., feature transformation $\mathbf{W}^{(l-1)}$ and nonlinear activation σ). The experimental results demonstrate that the simplified version performs better than the conventional GCN. In order to capture transition relationships between distant items, we stack multiple layers of T-GCN to aggregate high-order neighboring information of items. Formally, the information propagation process can be formalized as follows:

$$\mathbf{h}_i^{(l)} = \mathbf{A}_i^I \mathbf{H}^{(l-1)} + \mathbf{A}_i^S \mathbf{H}^{(l-1)} + \mathbf{A}_i^O \mathbf{H}^{(l-1)}, \tag{5}$$

where $\mathbf{A}_i^I, \mathbf{A}_i^O, \mathbf{A}_i^S$ are the ith row of the incoming matrix, outgoing matrix and self-connection matrix, respectively, l denotes the lth layer of T-GCN. After a L-layer T-GCN, we obtain the $\mathbf{h}_i^{(L)} \in \mathbb{R}^d$ which is the representation of node v_i.

In order to alleviate the problem of over-smoothing caused by stacking too many T-GCN layers, we further apply a HighWay Network, which merges the representation of item v_i after the L-layer T-GCN (i.e., $\mathbf{h}_i^{(L)}$) and its initial embedding (i.e., $\mathbf{h}_i^{(0)}$). Formally, the process is defined as follows:

$$\mathbf{g} = \sigma(\mathbf{W}_g([\mathbf{h}_i^{(L)} \| \mathbf{h}_i^{(0)}])), \tag{6}$$

$$\widetilde{\mathbf{h}}_i^{(L)} = \mathbf{g} \odot \mathbf{h}_i^{(L)} + (1 - \mathbf{g}) \odot \mathbf{h}_i^{(0)}, \tag{7}$$

where $\mathbf{W}_g \in \mathbb{R}^{d \times 2d}$ are trainable parameters, σ is the activation function *Sigmoid*. After the HighWay Network, we obtain the item representations $\widetilde{\mathbf{H}}^{(L)} = (\widetilde{\mathbf{h}}_1^{(L)}, \widetilde{\mathbf{h}}_2^{(L)}, \ldots, \widetilde{\mathbf{h}}_n^{(L)})$.

3.4. Temporal Interest Attention Network (TIAN)

In this subsection, we present the Temporal Interest Attention Network (TIAN), which further enhances item representations via capturing the similar user interest pattern of items within a session based on the temporal information. The basic assumption is that items within a short time interval may have similar user interest pattern. To the end, we separate items into different bins based on the time interval, i.e., items with close time interval are

separated into the same bin. It is worth noting that we separate items into different bins along the time dimension since different bins correspond to different user interest. Then we map each bin into an embedding and concatenate it with its corresponding item embedding. A novel Asymmetric Bidirectional Gated Recurrent Unit (Asym-BiGRU) is further developed to model the sequential relationships of bins within a session and obtain the enhanced representation of user interest bin sequence. At last, the enhanced user interest bin representations will be merged into their corresponding item representations with an attention network in order to enhance item representations with user interest information, and get the new representations of items.

3.4.1. User interest bin

In this subsection, we present how to separate items within a session into different user interest bins. A straightforward solution is that we employ a fixed splitting time span μ for all sessions and split items into different bins according to the their time interval to the last item in the session. Since different users may have distinct interest sensitivity, and also users' recent interacted items usually are more important for recommendation than historical ones, it would be inappropriate for utilizing a fixed splitting time span. To solve the issue, we propose to develop an adaptive time span to separate items into different user interest bins. The basic idea is that for items with small time intervals to the last item, our model will pay more attention to them and assigns a small time span to these items. Here a small time span indicates that our model is more sensitive to the time interval changes, i.e., a small time interval change of items will make them be mapped into different interest bins. On the contrary, for items with large time intervals to the last item, our model is less sensitive to them as they are distant from the user current interest, and will have small influence for the recommendation. It is worth noting that the time interval here is different to the one used in the TES-Graph. Specifically, the time interval utilized in TES-Graph is designed to infer user interest drift of a transition relationship, while time interval here is used to identify common user interests within a time span.

For a session $S = (v_1, v_2, \ldots, v_n)$, we denote $T = (t_1, t_2, \ldots, t_n)$ as the corresponding timestamp sequence of S, and $Q = (q_1, q_2, \ldots, q_n)$ as the time interval sequence where q_i is the time interval between the item v_i and the last item v_n (i.e., $q_i = t_n - t_i$). We first adopt a negative exponential function, and map the time interval sequence Q to an interest sensitivity sequence $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ as follows:

$$\gamma_i = e^{-\alpha(q_i + l)}, \tag{8}$$

where α and l are the decay constant and left offset, respectively. Note that an item v_i which is distant from the last item v_n will correspond to a small interest sensitivity value (i.e., γ_i is small), and vice versa. Then, we define the adaptive time span μ as follows:

$$\mu = \frac{e^{-\alpha l} - min(\Gamma)}{M}, \tag{9}$$

where M is a pre-defined number of time bins, which is determined by the average session length of the dataset. μ represents the desired adaptive time interval with respect to each session S. Next, the user interest bin of the item v_i is obtained as follows:

$$bin_i = k, \text{ where } \gamma_i \in (\mu \times (k-1), \mu \times k], \tag{10}$$

where $k \in (1, 2, \ldots, M)$. At last, we get the user interest bins $B = (bin_1, bin_2, \ldots, bin_n)$.

3.4.2. Asymmetric Bidirectional Gated Recurrent Unit (Asym-BiGRU)

After we obtain the user interest bins for each session, we embed the user interest bins sequence $B = (bin_1, bin_2, \ldots, bin_n)$ into an embedding sequence $\mathbf{E} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$, where $\mathbf{e}_i \in \mathbb{R}^d$ is a d-dimensional vector.

In order to enhance the representation of each bin, we propose to model the sequential relationships of bins within a session. A straightforward solution is to leverage a bidirectional gated recurrent unit (BiGRU) [38] to capture its forward and backward sides of contextual information. However, as the forward and backward sides of contextual information of a bin would play different roles (i.e., the forward contextual information would play a more important role as compared with the backward contextual information), treating them equally like in BiGRU would lead to a suboptimal bin representation. Thus, we propose a variant of BiGRU, named Asymmetric Bidirectional Gated Recurrent Unit (Asym-BiGRU). In Asym-BiGRU, the forward contextual information and the backward contextual information are treated asymmetrically and merged by the gate mechanism. At time-step i, the hidden state can be updated as follows:

$$\mathbf{e}_i = dropout(\mathbf{e}_i), \tag{11}$$

$$\overrightarrow{\mathbf{e}}_i = GRU(\overrightarrow{\mathbf{e}}_{i-1}, \mathbf{e}_i), \tag{12}$$

$$\overleftarrow{\mathbf{e}}_i = GRU(\overleftarrow{\mathbf{e}}_{i+1}, \mathbf{e}_i), \tag{13}$$

$$\mathbf{b}_{gate} = \sigma(\mathbf{W}_1[\overrightarrow{\mathbf{e}}_i \| \overleftarrow{\mathbf{e}}_i]), \tag{14}$$

$$\hat{\mathbf{e}}_i = \overrightarrow{\mathbf{e}}_i \odot \mathbf{b}_{gate} + (1 - \mathbf{b}_{gate}) \odot \overleftarrow{\mathbf{e}}_i, \tag{15}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times 2d}$ is trainable parameters, σ is the activation function *Sigmoid*, the *dropout* operation is used to alleviate overfitting. We also try other regularization methods [39,40] and the results are comparable to Dropout in this task. As the discussion of the regularization methods is out of the scope of our work, we leave it for the future work. Then, we obtain the enhanced representation of user interest bin sequence $\hat{\mathbf{E}} = (\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \ldots, \hat{\mathbf{e}}_n)$.

3.4.3. Attention network

After we get the enhanced representations of user interest bins $\hat{\mathbf{E}}$, we fuse them with their corresponding item representations $\widetilde{\mathbf{H}}^{(L)}$ with an attention network, which is defined as follows:

$$\beta_i = \mathbf{v}_3^T tanh(\mathbf{W}_2[\widetilde{\mathbf{h}}_i^{(L)} \| \hat{\mathbf{e}}_i]), \tag{16}$$

$$\mathbf{c}_i = \beta_i \widetilde{\mathbf{h}}_i^{(L)}, \tag{17}$$

where $\mathbf{W}_2 \in \mathbb{R}^{d \times 2d}$, $\mathbf{v}_3 \in \mathbb{R}^d$ are trainable parameters. The new representations of items in the session S are $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n)$ which capture both the sequential structural information of items and temporal user interests of a session.

3.5. Recommendation

Once we obtained the representations of items within the current session, i.e., $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n)$, we represent the long-term representation of the session $\mathbf{z}_{long}$ by combining all items' representations in the session. Similar to [27], we apply the sum-pooling as the combining function, which is defined as follows:

$$\mathbf{z}_{long} = \sum_{i=1}^{n} \mathbf{c}_i. \tag{18}$$

Meanwhile, we employ the GRU to get the short-term representation of the session:

$$\overrightarrow{\mathbf{c}}_i = GRU(\overrightarrow{\mathbf{c}}_{i-1}, \mathbf{c}_i), \tag{19}$$

where $\overrightarrow{\mathbf{c}}_{i-1}$ denotes the hidden state at time step $i-1$. Then, we use the representation of the last item as the short-term representation of the session, i.e., $\mathbf{z}_{short} = \overrightarrow{\mathbf{c}}_n$.

Finally, to generate the final session representation $\mathbf{z}_{final}$, we employ the gate mechanism to merge information from both long-term and short-term session representations, i.e., $\mathbf{z}_{long}$ and $\mathbf{z}_{short}$. Formally,

$$\mathbf{f} = \sigma(\mathbf{W}_3(\mathbf{z}_{long} \oplus \mathbf{z}_{short}) + \mathbf{b}_3), \tag{20}$$

$$\mathbf{z}_{final} = \mathbf{f} \odot \mathbf{z}_{long} + (1 - \mathbf{f}) \odot \mathbf{z}_{short}, \tag{21}$$

where $\mathbf{W}_3 \in \mathbb{R}^{d \times d}$, $\mathbf{b}_3 \in \mathbb{R}^d$ are trainable parameters, σ is the activation function *Sigmoid*.

Based on the final session representation $\mathbf{z}_{final}$, we produce the top-K recommendations from all candidate items in V. The recommendation probability of each candidate item $v_i \in V$ is determined by its initial embedding $\mathbf{h}_{v_i}$ and the final session representation $\mathbf{z}_{final}$. In particular, we use a dot product to calculate the recommendation score of each candidate item v_i, which is defined as:

$$\tilde{y}_i = \mathbf{z}_{final}^T \mathbf{h}_{v_i}. \tag{22}$$

Then we apply a softmax function over all scores to obtain the probability distribution of all candidate items $\hat{y}$:

$$\hat{y} = \text{softmax}(\tilde{y}), \tag{23}$$

where $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_{|V|})$, and $\hat{y}_i$ denotes the probability of item $v_i \in V$ appearing as the next-click in the current session items.

3.6. Objective function

Since our goal is to maximize the prediction probability of the actual item given the current session, we apply the cross-entropy as the loss function:

$$\mathcal{L} = -\sum_{i=1}^{|V|} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i), \tag{24}$$

where y denotes the one-hot encoding vector of the ground truth. We leverage the back-propagation through time (BPTT) algorithm to train our proposed model.

4. Experiments

In this section, we aim to answer the following six research questions and carry out extensive experiments to estimate the performance of our proposed approach TE-GNN :

- **RQ1**: Does TE-GNN outperform the state-of-the-art session-based recommendation baselines for the session-based recommendation task?
- **RQ2**: Does the Temporal Graph Convolutional Network (T-GCN) influence the performance of TE-GNN?
- **RQ3**: Does the Temporal Interest Attention Network (TIAN) affect the performance of TE-GNN?
- **RQ4**: What are the influence of the number of Temporal Graph Convolutional Network (T-GCN) layers on the model performance?
- **RQ5**: How well does TE-GNN perform on sessions with different lengths?
- **RQ6**: What is the computational complexity of proposed TE-GNN?

4.1. DataSets

We employ four widely used benchmark datasets, including *Diginetica*, *Tmall*, *Nowplaying* and *Retailrocket* to evaluate the performance of TE-GNN.

Table 1
Statistics of datasets used in the experiment.

Dataset	Diginetica	Tmall	Nowplaying	Retailrocket
#click	982 961	818 479	1 367 963	2 756 101
#train	719 470	351 268	825 304	175 325
#test	60 858	25 898	89 824	24 283
#unique items	43 097	40 728	60 417	22 305
#Average length	5.12	6.69	7.42	3.96

- **Diginetica**.[1] This dataset is released from the CIKM Cup 2016. Because of its transaction data, it is often used in session-based recommendation task. Following [15,17,27,37], we extract the data of the last week as the test set and use the remainder for training.
- **Tmall**.[2] It is extracted from the IJCAI-15 competition, which is made up of anonymous users' shopping logs on Tmall online shopping platform. As the number of items is too large, following [17], we select the portions 1/64 of sessions as the dataset, in which sessions of the last day are used as test data, the remaining sessions are leveraged for training.
- **Nowplaying**.[3] This dataset is built from Twitter which describes the music listening habits of users [41]. Similar to [17], we split the sessions for training and test, where the sessions of last two months are utilized for test and the remaining historical data as the training set.
- **Retailrocket**.[4] It is released by the Kaggle competition 2016, and is collected from a real world e-commerce website. Retailrocket includes the behavior data of website visitors within 4.5 months, and the behavior of visitors are divided into three categories: click, join shopping cart and transaction. We select the most recent portions of 1/4 sessions as the dataset, where sessions of last 15 days are employed as the test set and the remaining data for training.

Following [15,17], we filter out sessions of length 1 and items appearing less than 5 times. We also utilize sequence splitting pre-processing to generate sequences as well as their corresponding labels. Taking the session sequence $\mathcal{S} = (v_1, v_2, v_3, \ldots, v_n)$ as an example, we generate sequences and labels as $([v_1], v_2)$, $([v_1, v_2], v_3)$, $\ldots$, $([v_1, \ldots, v_{n-1}], v_n)$ for training and testing data. The statistics of datasets are illustrated in Table 1.

4.2. Evaluation metrics

Following previous works [15,17,27,35], we adopt two widely used metrics, i.e., *P@K* and *MRR@K* to evaluate the performance of our model.

- *P@K*(Precision): It measures the proportion of cases when the target item is ranked within the top-K recommendations. *P@K* is a metric which is used to evaluate unranked results.

  $$P@K = \frac{n_{hit}}{N}, \tag{25}$$

  where N indicates the number of test cases and n_{hit} is the number of cases that the target item is in the top-K items of the ranked list.
- *MRR@K*(Mean Reciprocal Rank): It is the average of the reciprocal ranks of the target item in the recommendation list.

[1] https://competitions.codalab.org/competitions/11161.
[2] https://tianchi.aliyun.com/dataset/dataDetail?dataId=42.
[3] http://dbis-nowplaying.uibk.ac.at/#nowplaying.
[4] https://www.kaggle.com/retailrocket/ecommerce-dataset.

This metric considers the position of correct recommended items in a ranked list.

$$MRR@K = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}, \tag{26}$$

where N is the number of test cases and $rank_i$ is the position of the ith target item in the list of recommended items. Note that if the target item is not in the top-K items, its $MRR@K$ score is set to 0.

Note that the $K = 20$ and a higher value of $P@K$ and $MRR@K$ indicate a better model performance.

4.3. Baselines

To evaluate the performance of our model comprehensively, we compare it with eleven baseline methods which can be grouped into three categories, i.e., traditional recommendation methods, RNN and attention based methods, graph neural network based methods. The details of all baselines are briefly described as follows:

Traditional recommendation methods:

- POP: This is a frequently used baseline method in recommendation system, which recommends the top-N most frequent items in the training set.
- Item-KNN [3]: This method recommends items which are most similar to the current item in the session, where similarity is defined as the co-occurrence relationship between items.
- FPMC [1]: It is a hybrid method for sequential recommendation based on Markov chain. The user representation is ignored as it is not available in the scenario of session-based recommendation.

RNN and attention based methods

- GRU4REC [5]: This method applies the GRU for session-based recommendation. It modifies the basic GRU by introducing session-parallel mini-batches, mini-batch based output sampling and ranking loss function.
- NARM [8]: It employs RNN to model the user's sequential behavior and captures a user's main preference with the attention mechanism. The recommendation probability for each candidate item is computed by a bi-linear matching scheme based on the unified session representation.
- STAMP [9]: This method attempts to address the user interests drift issue by capturing both a user's general interests from the long-term memory of a session and her current interests from the short-term memory of the last-clicks.
- CSRM [26]: This approach is composed of an inner memory encoder and an outer memory encoder. The former is used to model a user's own information in the current session via RNN and attention mechanism, while the latter is designed to capture the intent of the current session by exploiting collaborative information from neighborhood sessions.
- SR-IEM [27]: This method utilizes an improved attention mechanism to generate item's importance score, and generates session representation based on a user's global preference and her current interest.

Graph neural network based methods:

- SR-GNN [15]: SR-GNN attempts to capture the complex transitions of items by modeling session sequences as graph structured data. It employs the gated graph neural networks to obtain the representation of items and combines a self attention mechanism to generate session representation.
- TAGNN [35]: It captures the complex item transitions within sessions by modeling items in sessions as session graphs and obtains item embeddings using graph neural networks. It also introduces a target attentive module to reveal the relevance of historical actions given a target item to improve the session representations.
- GCE-GNN [17]: It is a state-of-the-art GNN-based model, which learns item representations from two different levels, i.e., session-level and global-level. The session-level item representation aims to model pairwise item-transitions within the current session while the global-level item representation attempts to model pair-wise item-transitions over all sessions.

4.4. Parameter setting

In our implementation, we choose the training mini-batch as 256 and the embedding dimension of item as 256 in all experiments. We set the number of interest blocks M to 6 in Eq. (9), and the number of T-GNN layers to 2. Following [16,17,27,35], all parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. The initial learning rate for Adam is set to 0.001 and will decay by 0.1 after every 3 training epochs. The L_2 penalty is set to 10^{-5} and the dropout rate is set to 0.3 to avoid overfitting.

4.5. Overall comparison (RQ1)

To verify the effectiveness of TE-GNN, we report the performance comparison between TE-GNN and all baseline methods in Table 2. From Table 2, we can observe that Item-KNN [3] mostly achieves the best performance on all datasets among all traditional recommendation methods. This is attributed to that the potential preference of users have an important impact on the effect of recommendation.

Comparing with the traditional recommendation methods, RNN and attention based methods usually demonstrate a superior performance. In particular, GRU4REC, which is the first RNN based approach for the task of session based recommendation, performs better than Item-KNN on Tmall and Retailrocket, while performs worse than it on Diginetica and Nowplaying. Both NARM and STAMP perform better than GRU4REC and Item-KNN as they further incorporate the attention mechanism in order to capture the importance of items in session dynamically. CSRM [26] usually achieves a better performance than NARM and STAMP on all datasets as it incorporates auxiliary information from other sessions to enhance the current session representation. SR-IEM [27] obtains a competitive performance as that of CSRM. It modifies the self-attention mechanism to obtain a better user's long-term preference and then it combines user's long-term preference and short-term preference conveyed by the last item in the session to make recommendation.

Among all baseline methods, these graph neural network (GNN) based methods usually show a superior performance. The main reason may be that GNN slightly relaxes the assumption of temporal dependence between consecutive items and models more complex user item transitions as pairwise relations (e.g., directed graph). For example, SR-GNN [15] attempts to capture more implicit connections between items in a session and models session sequences as graph-structured data. TAGNN [35] further takes into account user preferences with target-aware attentions. Among all GNN-based methods, GCE-GNN [17] mostly presents the best performance on all datasets, as it can effectively learn item representations from both global context and local context.

Our proposed approach TE-GNN shows superiority over all state-of-the-art baseline methods on all datasets. Specifically, the

Table 2
Comparison of the effects of each model on four datasets. The best results in terms of the corresponding metric are highlighted in bold.

Dataset	Diginetica		Tmall		Nowplaying		Retailrocket	
Methods	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20
POP	1.18	0.28	2.00	0.90	2.28	0.86	1.97	0.75
Item-KNN	35.75	11.57	9.15	3.31	15.94	4.91	10.23	3.96
FPMC	22.14	6.66	16.06	7.32	7.36	2.82	9.65	4.32
GRU4REC	30.79	8.22	10.93	5.89	7.92	4.48	41.35	25.54
NARM	48.32	16.00	23.30	10.42	18.59	6.93	59.46	41.48
STAMP	46.62	15.13	26.47	13.36	17.66	6.88	58.48	38.96
CSRM	50.55	16.38	29.46	13.96	18.14	6.08	61.09	40.28
SR-IEM	52.35	17.64	23.98	11.03	19.85	8.22	59.85	38.92
SR-GNN	51.26	17.78	27.57	13.72	18.87	7.47	60.19	39.64
TAGNN	51.31	18.03	33.75	15.91	18.65	7.15	59.31	39.65
GCE-GNN	54.22	19.04	33.42	15.42	22.37	**8.40**	63.29	40.35
TE-GNN	**54.78***	**19.35***	**39.01***	**18.47***	**23.85***	7.88	**65.02***	**44.02***

*Indicates that TE-GNN significantly outperforms the best baseline GCE-GNN at p-value < 0.01 using the t-test.

Table 3
Performance comparison of each GCN.

Datasets	Diginetica		Tmall		Nowplaying		Retailrocket	
Methods	P@20	M@20	P@20	M@20	P@20	M@20	P@20	M@20
TE-GNN-MLP	51.43	15.79	29.36	13.83	20.38	**8.42**	61.27	38.81
TE-GNN-GGNN	51.36	17.65	33.94	16.62	21.35	8.18	63.57	44.11
TE-GNN-GAT	54.45	18.98	38.95	17.88	23.61	7.41	63.81	41.46
TE-GNN	**54.78**	**19.35**	**39.01**	**18.47**	**23.85**	7.88	**65.02**	**44.02**

performance improvements of TE-GNN over the best performing baseline GCE-GNN on Diginetica, Tmall, Nowplaying, Retailrocket in terms of P@20 are 1.03%, 16.73%, 6.62%, 2.73%, respectively. Similar performance improvements can also be observed in terms of MRR@20. The results reveal the effectiveness to model the complex user interest drift pattern and user common interest pattern. In addition, TE-GNN can effectively model both pattern information by introducing the Temporal Graph Convolutional Network (T-GCN) and the Temporal Interest Attention Network (TIAN).

4.6. Impact of Temporal Graph Convolutional Network (T-GCN) (RQ2)

To investigate the impact of the Temporal Graph Convolutional Network (T-GCN), we compare it with several variants, including:

- TE-GNN-MLP: We replace the T-GCN component in TE-GNN with a multilayer perceptron (MLP).
- TE-GNN-GGNN: It uses the gated graph neural network (GGNN) [12,15] to replace the T-GCN in TE-GNN.
- TE-GNN-GAT: It replaces the T-GCN in TE-GNN with the graph attention network (GAT) [17,42].

Table 3 shows that all three variants mostly lead to a considerable performance degradation on all four datasets in term of both metrics. More precisely, replacing T-GCN with MLP demonstrates the worst performance as it lacks the capability to capture the complex user item transition information. Both TE-GNN-GGNN and TE-GNN-GAT show a superior performance to TE-GNN-MLP as they effectively model the rich structure information between items based on a graph neural network. Among them, TE-GNN-GAT performs better than TE-GNN-GGNN which would be attributed to that it considers the importance of neighboring items via the graph attention network. Our proposed method TE-GNN achieves a superior performance to all three variants, which reveals the effectiveness of incorporating the Temporal Graph Convolutional Network (T-GCN) in our model.

4.7. Impact of Temporal Interest Attention Network (TIAN) (RQ3)

To answer RQ3, we compare the performance of Temporal Interest Attention Network (TIAN) with the self attention network and the position-aware attention network, which are utilized in SASRec [23] and SGNN [16], respectively. Different from both self attention network and position-aware attention network, the fine-grained temporal information are explored with the proposed Temporal Interest Attention Network. In addition, we further investigate the effectiveness of the proposed component Asymmetric Bidirectional Gated Recurrent Unit (Asym-BiGRU) in TIAN by discarding the Asym-BiGRU component or replacing it with the conventional BiGRU. A brief description of all variants are listed as follows:

- TE-GNN-w/o TIAN: We remove the Temporal Interest Attention Network (TIAN) in TE-GNN.
- TE-GNN-SA: We replace the Temporal Interest Attention Network (TIAN) in TE-GNN with the self attention network.
- TE-GNN-PA: We adopt the position-aware attention network in SGNN [16] rather than the Temporal Interest Attention Network (TIAN) in TE-GNN.
- TIAN-w/o GRU: We discard the Asym-BiGRU in TIAN and do not consider the sequential relationships of bins within a session.
- TIAN-BiGRU: We replace the Asym-BiGRU with the conventional BiGRU in TIAN.

From Table 4, we can observe that incorporating TIAN will achieve a substantial performance improvement, which shows the effectiveness of TIAN. TE-GNN-w/o TIAN shows a significant performance degradation as it cannot model the user temporal interest information. Comparing to TE-GNN-w/o TIAN, both TE-GNN-SA and TE-GNN-PA do not demonstrate a substantial performance improvement. Since TE-GNN-SA only captures the semantic relationships among items and TE-GNN-PA focuses on model items' position information, both of them ignore the user

Table 4
Performance comparison of our proposed method and all variants.

Datasets	Diginetica		Tmall		Nowplaying		Retailrocket	
Methods	P@20	M@20	P@20	M@20	P@20	M@20	P@20	M@20
TE-GNN-w/o TIAN	52.47	18.21	35.96	17.13	21.65	7.31	54.83	35.42
TE-GNN-SA	52.14	18.12	32.35	15.18	21.29	7.15	55.03	35.98
TE-GNN-PA	52.53	18.46	32.25	15.28	22.13	7.35	53.54	35.97
TIAN-w/o GRU	53.42	18.89	37.25	17.73	22.74	7.56	60.21	39.66
TIAN-BiGRU	54.47	19.18	38.55	18.19	23.64	7.69	63.84	42.67
TE-GNN	**54.78**	**19.35**	**39.01**	**18.47**	**23.85**	**7.88**	**65.02**	**44.02**

Table 5
Analysis of the computational complexity of different comparing models. n is the session length, s denotes the number of graph neural networks layers, d is the dimension of item embedding, $|V|$ is the number of all candidate items, k is the number of GRU layers, and K is the number of global neighbors of item.

Methods	Complexity
SR-GNN	$O(s(nd^2+n^3)+d^2)$
TAGNN	$O(s(nd^2+n^3)+n\|V\|d^2+d^2)$
GCE-GNN	$O(sn^2d+nKd+d^2)$
TE-GNN	$O((s+k)n^2d+d^2)$

temporal interest information. Differ from these variants, TIAN-w/o GRU obtains a superior performance as it utilizes a multi-grain interest separating strategy to explicitly model user temporal interest information. It is worth noting that the performance drop of TIAN-w/o GRU is relatively smaller than that of the other three comparing methods (i.e., TE-GNN-w/o TIAN, TE-GNN-SA and TE-GNN-PA). The main reason is that the other three comparing methods choose to discard the important sub-module TIAN or replace it with a simple attention-based network (e.g., the self-attention network or the position-aware attention network), thus they have a considerable performance drop. However, in TIAN-w/o GRU, we still keep the sub-module TIAN and only remove the GRU part (i.e., Asym-BiGRU) from TIAN. TIAN-BiGRU boosts the performance by further taking the contextual information of each interest bin within a session into consideration. However, in the scenario of session-based recommendation, the forward contextual information would play a more important role as compared with the backward contextual information. Comparing with all variants, our proposed approach TE-GNN achieves the best performance by incorporating the Asym-BiGRU which models the two sides of contextual information differently.

4.8. Impact of the number of Temporal Graph Convolutional Network (T-GCN)

layers L (RQ4) In order to check the impact of the number of Temporal Graph Convolutional Network (T-GCN) layers in TE-GNN, we study the performance of TE-GNN with various number of T-GCN layers ranging from 1 to 5. Meanwhile, we also compare the performance of our method with its corresponding variant TE-GNN-w/o HN, which discards the HighWay Network, on different number of T-GCN layers. The results on all datasets are demonstrated in Fig. 4. We can observe that on the Diginetica dataset the performance of TE-GNN first rises up and reaches the peak when $L=2$, after that it starts to decrease if L becomes larger. Similar results are observed on both Nowplaying and Retailrocket. On the Tmall dataset, the performance of TE-GNN increases gradually and reaches the peak when $L=4$ in terms of both metrics. It will present a considerable performance degradation if we further enlarge the number of the layers. The trend demonstrates that our model can effectively capture transition relationships between distant items when we increase the number of T-GCN layers. However, if the number of T-GCN layers becomes too large, more irrelevant or noise high-order neighboring items will be injected into the model which leads to a sub-optimal performance. We can also observe in Fig. 4 that our model TE-GNN consistently performs better than its corresponding variant TE-GNN-w/o HN. In addition, with the increase of number of T-GCN layers, the performance degradation of TE-GNN-w/o HN as comparing to our method TE-GNN becomes larger. The results verify that incorporating the HighWay Network can effectively alleviate the over-smoothing issue caused by these multiple layers of GCN architecture.

4.9. Impact of the session length (RQ5)

To investigate the performance of our method on sessions with different lengths, we separate all sessions into two groups, i.e., short sessions and long sessions, based the session length. Specifically, the sessions with length larger than 5 are considered as long sessions and the remaining sessions are considered as short sessions. We also report the performance of the two best performing baselines, i.e., TAGNN [35] and GCE-GNN [17]. Fig. 5 demonstrates the performance of three methods with respect to different session lengths. From the results, we have the following observations: Firstly, on both Diginetica and Retailrocket datasets, all methods demonstrate a relatively higher performance on the short sessions than that on the long sessions. However, on both Tmall and Nowplaying datasets, all methods show a higher performance on the long sessions than that on the short sessions. The reason may be that both Diginetica and Retailrocket datasets have a smaller session length as compared with that of Tmall and Nowplaying datasets. Secondly, our proposed method TE-GNN is consistently better than both TAGNN and GCE-GNN on both short and long sessions for all datasets with an exception of Nowplaying in terms of the metric MRR@20.

4.10. Comparison of computational complexity (RQ6)

To analyze the computational complexity of our proposed method TE-GNN, we compare it with three state-of-the-art baselines, i.e., SR-GNN, TAGNN, and GCE-GNN. Table 5 reports the results of theoretical analysis of the time complexity. For SR-GNN, the computational complexity is $O(s(nd^2+n^3)+d^2)$ where d and n are the dimension of item embeddings and the session length, respectively. Here, we use s to denote the number of layers in graph neural networks. TAGNN demonstrates a higher computational complexity, i.e., $O(s(nd^2+n^3)+n|V|d^2+d^2)$ where $|V|$ indicates the number candidate items, than that of SR-GNN. This is because TAGNN applies the GGNN [43] to learn node vector, which is similar to SR-GNN. In addition, it also adopts a local target attentive module to measure attention scores between each item in the current session and all items in V. The computational complexity of GCE-GNN is $O(sn^2d+nKd+d^2)$, where K is the number of global neighbors of an item. For our proposed method TE-GNN, the computational complexity is $O((s+k)n^2d+d^2)$, where k is the number of GRU layers. From the results, we can observe that the computational complexity of TE-GNN is much lower than that of TAGNN, and comparable to that of SR-GNN and GCE-GNN.

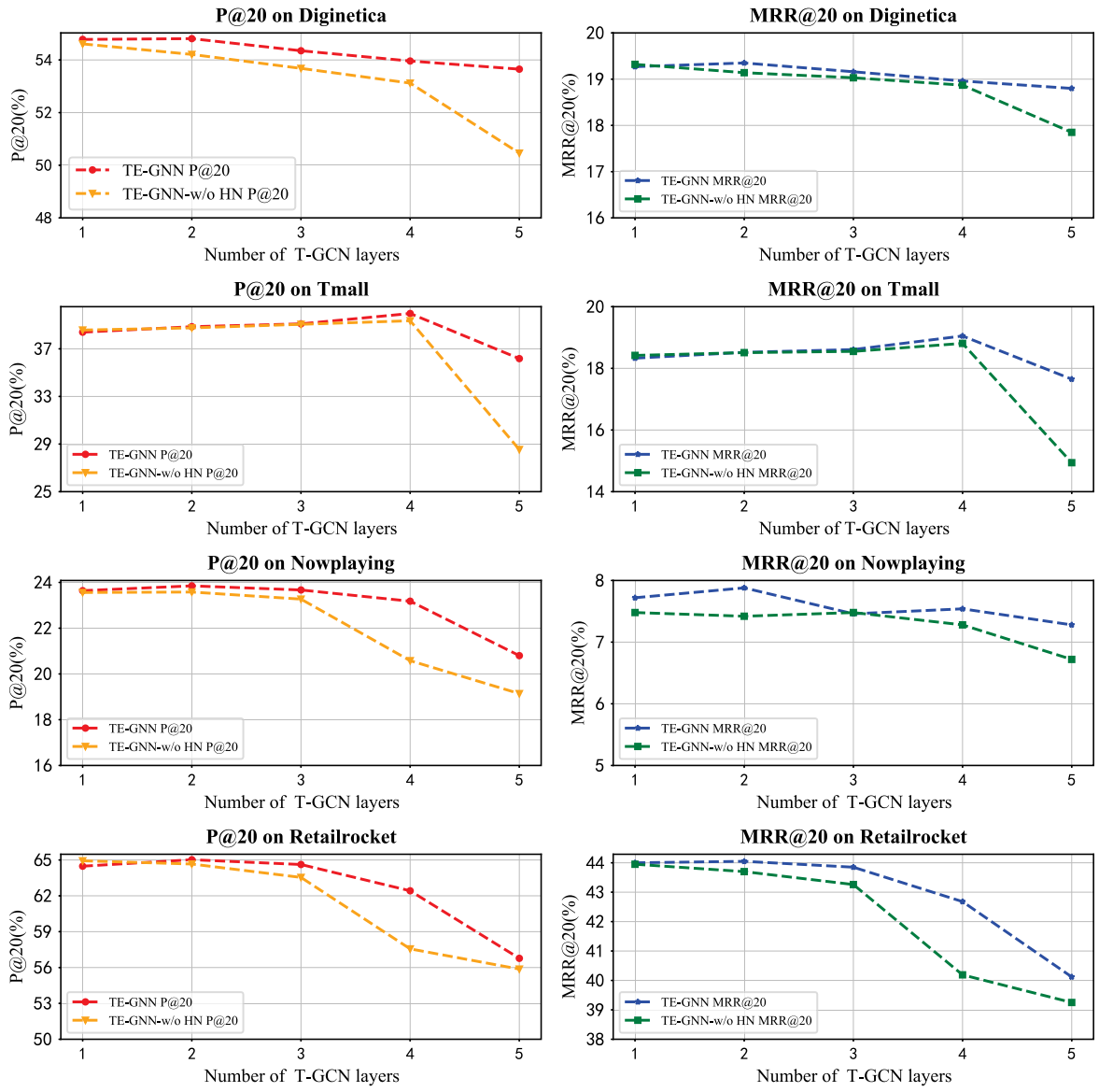

Fig. 4. Impact of the number of T-GCN layers L.

5. Conclusion

This work investigates the problem of session-based recommendation by effectively modeling the complex user interest shift pattern. To this end, we extend the conventional session graph by incorporating the user interest drift and propose a novel time-enhanced session graph (TES-Graph). Based on the TES-Graph, a Temporal Graph Convolutional Network (T-GCN) is designed to learn better item embeddings. In addition, we also introduce a Temporal Interest Attention Network (TIAN) to model items which share a similar user interest along the temporal dimension. Extensive experiments on four datasets show that our proposed approach TE-GNN consistently outperforms all state-of-the-art baseline methods.

Existing works mainly focus on exploring local information within the current session for recommendation, while rich global information from other neighboring sessions are largely ignored. In addition, there are noisy transition information between items caused by users' accidental or wrong clicks, which may be injected into the model training process. In the future, we will develop a more effective model by incorporating the rich information from a global perspective, and dealing with noisy items in order to alleviate the noisy transition issue.

CRediT authorship contribution statement

Gu Tang: Conceptualization of this study, Methodology, Software, Writing – original draft. **Xiaofei Zhu:** Writing – original draft & review & editing, Supervision. **Jiafeng Guo:** Writing – review & editing, Supervision. **Stefan Dietze:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China [No. 62141201] and the Federal Ministry of Education and Research [No. 01IS21086].

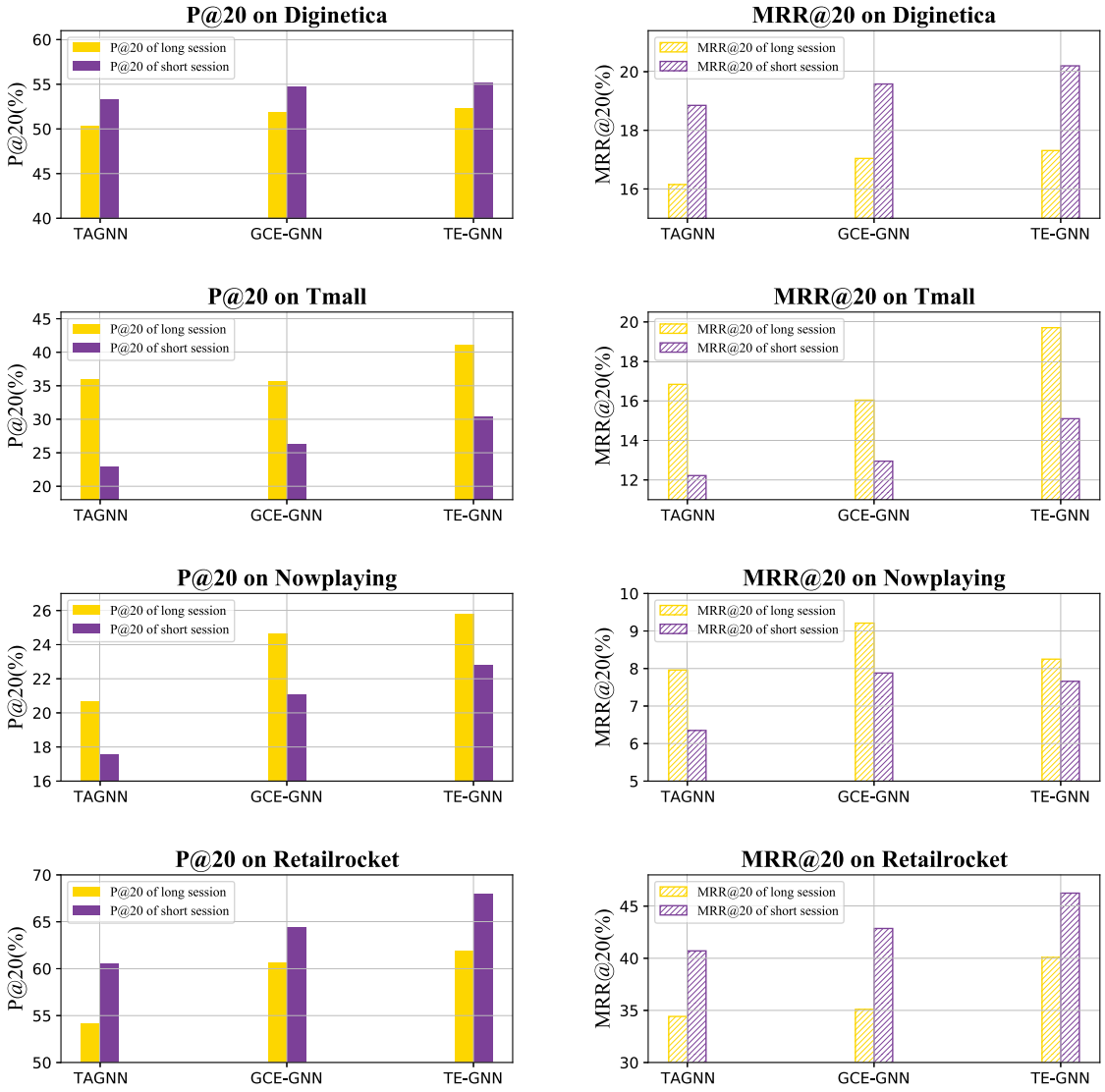

Fig. 5. Impact of the session length.

References

[1] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized Markov chains for next-basket recommendation, in: Proceedings of the 19th International Conference on World Wide Web, (2010), pp. 811–820.

[2] G. Shani, D. Heckerman, R.I. Brafman, An MDP-based recommender system, J. Mach. Learn. Res. 6 (2005) 1265–1295.

[3] B.M. Sarwar, G. Karypis, J.A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the Tenth International World Wide Web Conference, (2001), pp. 285–295.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua, Neural Collaborative Filtering, in: Proceedings of the 26th International Conference on World Wide Web, (2017), pp. 173–182.

[5] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based Recommendations with Recurrent Neural Networks, in: Proceedings of the 4th International Conference on Learning Representations, (2016).

[6] B. Hidasi, A. KaratzoglouBalázs, 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), pp. 843–852.

[7] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, T.-Y. Liu, Sequential click prediction for sponsored search with recurrent neural networks, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, (2014).

[8] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, J. Ma, Neural Attentive Session-based Recommendation, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, (2017), pp. 1419–1428.

[9] Q. Liu, Y. Zeng, R. Mokhosi, H. Zhang, STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (2018), pp. 1831–1839.

[10] K. Cho, B.V. Merrienboer, C. Gulcehre, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, (2014), pp. 1724–1734.

[11] T.N. Kipf, M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, in: Proceedings of the 5th International Conference on Learning Representations, (2017).

[12] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated Graph Sequence Neural Networks, in: Proceedings of the 2016 International Conference on Learning Representations, (2016).

[13] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2009) 61–80.

[14] K. Sun, T. Qian, T. Chen, Y. Liang, Q.V.H. Nguyen, H. Yin, Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, (2020), pp. 214–221.

[15] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-Based Recommendation with Graph Neural Networks, in: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, 2019, (2019), pp. 346–353.

[16] Z. Pan, F. Cai, W. Chen, H. Chen, M. de Rijke, Star Graph Neural Networks for Session-based Recommendation, in: Proceedings of the 29th ACM International Conference on Information and Knowledge Management, (2020), pp. 1195–1204.

[17] Z. Wang, W. Wei, G. Cong, X.-L. Li, X.-L. Mao, M. Qiu, Global Context Enhanced Graph Neural Networks for Session-based Recommendation, in:

Proceedings of the 43nd International ACM SIGIR Conference on Research and Development in Information Retrieval, (2020), pp. 169–178.
[18] Y. Chen, S. Mensah, F. Ma, H. Wang, Z. Jiang, Collaborative filtering grounded on knowledge graphs, Pattern Recognit. Lett. 151 (2021) 55–61.
[19] L. Li, W. Pan, Z. Ming, Cofi-points: Collaborative filtering via pointwise preference learning on user/item-set, ACM Trans. Intell. Syst. Technol. 11 (4) (2020) 41:1–41:24.
[20] W. Chen, C. Fei, H. Chen, M.D. Rijke, Joint neural collaborative filtering for recommender systems, in: ACM Transactions on Information Systems(2019), TOIS, 2019, pp. 39:1–39:30.
[21] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, in: ACM Transactions on Information Systems(2004), TOIS, 2004, pp. 5–53.
[22] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, X. Cheng, Learning Hierarchical Representation Model for NextBasket Recommendation, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, (2015), pp. 403–412.
[23] W. Kang, J. McAuley, Self-Attentive Sequential Recommendation, in: Proceedings of the 2018 IEEE International Conference on Data Mining, (2018), pp. 197–206.
[24] Q. Zhang, B. Wu, Z. Sun, Y. Ye, Gating augmented capsule network for sequential recommendation, Knowl.-Based Syst. 247 (2022) 108817.
[25] J. Song, H. Shen, Z. Ou, J. Zhang, T. Xiao, S. Liang, ISLF: Interest Shift and Latent Factors Combination Model for Session-based Recommendation, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence Main Track, (2019), pp. 5765–5771.
[26] M. Wang, P. Ren, L. Mei, Z. Chen, J. Ma, M. de Rijke, A Collaborative Session-based Recommendation Approach with Parallel Memory Modules, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, (2019), pp. 345–354.
[27] Z. Pan, F. Cai, Y. Ling, M. de Rijke, Rethinking Item Importance in Session-based Recommendation, in: Proceedings of the 43nd International ACM SIGIR Conference on Research and Development in Information Retrieval, (2020), pp. 25–30.
[28] L. Liu, L. Wang, T. Lian, CaSe4SR: Using category sequence graph to augment session-based recommendation, Knowl.-Based Syst. 212 (2021) 106558.
[29] H. Tang, G. Zhao, X. Bu, X. Qian, Dynamic evolution of multi-graph based collaborative filtering for recommendation systems, Knowl.-Based Syst. 228 (2021) 107251.
[30] J. Wang, K. Ding, Z. Zhu, J. Caverlee, Session-based recommendation with hypergraph attention networks, in: Proceedings of the 2021 SIAM International Conference on Data Mining, SIAM, 2021, pp. 82–90.
[31] H. Wang, Y. Zeng, J. Chen, Z. Zhao, H. Chen, A spatiotemporal Graph Neural Network for session-based recommendation, Expert Syst. Appl. 202 (2022) 117114.
[32] Z. Yang, S. Dong, J. Hu, GFE: General knowledge enhanced framework for explainable sequential recommendation, Knowl.-Based Syst. 230 (2021) 107375.
[33] C. Feng, C. Shi, S. Hao, Q. Zhang, X. Jiang, D. Yu, Hierarchical Social Similarity-guided Model with Dual-mode Attention for session-based recommendation, Knowl.-Based Syst. 230 (2021) 107380.
[34] N. Zhu, J. Cao, X. Lu, H. Xiong, Learning a hierarchical intent model for next-item recommendation, ACM Trans. Inform. Syst. (2021) 40 (2021) 38:1–38:28.
[35] F. Yu, Y. Zhu, Q. Liu, S. Wu, L. Wang, T. Tan, TAGNN: Target Attentive Graph Neural Networks for Session-based Recommendation, in: Proceedings of the 43nd International ACM SIGIR Conference on Research and Development in Information Retrieval, (2020), pp. 1921–1924.
[36] X. Zhu, J. Guo, X. Cheng, Y. Lan, More than relevance: high utility query recommendation by mining users' search behaviors, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, (2012), pp. 1814–1818.
[37] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation, in: Proceedings of the 43nd International ACM SIGIR Conference on Research and Development in Information Retrieval, (2020), pp. 639–648.
[38] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, (2014), pp. 1724–1734.
[39] Y. Rong, W. Huang, T. Xu, J. Huang, DropEdge: Towards Deep Graph Convolutional Networks on Node Classification, in: Proceedings of the 8th International Conference on Learning Representations, (2020).
[40] H. Zhang, M. Xu, G. Zhang, K. Niwa, SSFG: Stochastically scaling features and gradients for regularizing graph convolutional networks, 2021, arXiv preprint arXiv:2102.10338.
[41] E. Zangerle, M. Pichl, W. Gassler, G. Specht, #nowplaying Music Dataset: Extracting Listening Behavior from Twitter, in: Proceedings of the First International Workshop on Internet-Scale Multimedia Management, (2014), pp. 21–26.
[42] P. Veliković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: Proceedings of the 2018 International Conference on Learning Representations, (2018).
[43] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated Graph Sequence Neural Networks, in: Proceedings of the 4th International Conference on Learning Representations, ICLR, (2016).