Neural Computing and Applications (2023) 35:13077–13092
https://doi.org/10.1007/s00521-021-05723-2

S.I. : DEEP SOCIAL COMPUTING

Item trend learning for sequential recommendation system using gated graph neural network

Ye Tao[1] · Can Wang[1] · Lina Yao[2] · Weimin Li[3] · Yonghong Yu[4]

Received: 12 July 2020 / Accepted: 7 January 2021 / Published online: 6 February 2021

Abstract

Recommendation system, or recommender system, is widely used in online Web applications like e-commerce Web sites and movie review Web sites. Sequential recommender put more emphasis upon user's short-term preference through exploiting information from its recent history. By incorporating the user short-term preference into the recommendation, the algorithm achieves a higher accuracy, which proves that a more accurate user portrait or representation boosts the performance to a great extent. Intuitionally, we seek to improve the current item representation modeling via incorporating the item trend information. Most of the recommendation models neglect the importance of the ever-changing item popularity. To this end, this paper introduces a novel sequential recommendation approach dubbed TRec. TRec learns the item trend information from the implicit user interaction history and incorporates the item trend information into the subsequent item recommendation tasks. After that, a self-attention mechanism is used for better representation. We also investigate alternative ways to model the proposed item trend representation; we evaluate two variant models which leverage the power of gated graph neural network upon the item trend representation modeling to boost the representation ability. We conduct extensive experiments with seven baseline methods on four benchmark datasets. The empirical results show that our proposed approach outperforms the state-of-the-art models as high as 18.2%. The experiment result displays the effectiveness in item trend information learning while with low computational complexity as well. Our study demonstrates the importance of item trend information in recommendation system.

Keywords Sequential recommendation · Information retrieval · Self-attention · Implicit interaction · Trend

✉ Can Wang
can.wang@griffith.edu.au

Ye Tao
ye.tao2@griffithuni.edu.au

Lina Yao
lina.yao@unsw.edu.au

Weimin Li
wmli@shu.edu.cn

Yonghong Yu
yuyh@njupt.edu.cn

[1] School of ICT, Griffith University, Gold Coast, Australia

[2] School of Computer Science, University of New South Wales, Sydney, Australia

[3] School of Computer Engineering and Science, Shanghai University, Shanghai, China

[4] Tongda College, Nanjing University of Posts and Telecommunications, Nanjing, China

1 Introduction

Recommendation systems have been integrated into most modern web applications and have greatly alleviated the pains that users have endured when facing enormous overloaded information via providing smart recommendation results to users. Most recommendation systems receive the user's past interaction history as input to generate the proper representation. The recommendation algorithm then feed these inputs into its prediction model to yield prediction scores and recommendation outcomes which are then offered to the front-end Web applications. Collaborative filtering-based (CF) [1] approaches and content-based (CB) approaches [2] are two groups of mainstream traditional recommendation methods. More recently, the latent factor-based models such as MF (matrix factorization) [3] are widely used for good prediction accuracy along with low runtime cost. Latent factor-based methods are able to conveniently model the representation of user long-term preference; however, many works [4, 5] have pointed out that such approaches fail to catch up with the user's ever-changing taste.

To solve this issue, recent research on sequential recommendation system incorporates the user short-term preference into the traditional user latent vector [6]; the user short-term preference can be derived from the recent user–item interaction history. Leveraging upon the mixed user long-term and short-term preferences, the prediction model is therefore able to comparably give a higher weight score on the recent user interaction behaviors. Nevertheless, most of these works neglect a similar fact: The extent to which the items are accepted and loved by the users is in a state of flux. Let us take the smart phone lithium battery charger for instance, recalling in the past years when the smart phone lithium battery charger is essential for every phone user; however, with the emergence of the smart phones with built-in batteries, an external phone battery charger is no longer popular though it is still on sale. It actually yet has nothing to do with the quality of the item itself. That is to say, we need to take this ever-changing popularity of items into account. Recent recommender systems tend to neglect such a shifting item popularity. For example, based on a former kung-fu lover's interaction history, a traditional factorization-based recommender might give a high prediction score on a target kung-fu movie. As a result, we can see that both the classical matrix factorization approaches [4, 5, 7] or the recent deep learning-based approaches [8, 9] fail to find a way to model items' changing popularity.

Aiming to deal with the aforementioned problems, we propose ways to model the item trend information; different approaches are used to achieve this goal. In addition to the one we proposed previously [10], different approaches for the item trend representation modeling with gated graph neural network are discussed as well. Graph neural network, as a powerful tool for the node embeddings learning, is utilized to generate the reliable and accurate item trend representation for candidate items; this technique is capable of providing rich local contextual information by encoding node features, while vanilla GNN is best for non-sequential input, for example, features like height, weight and gender of a person; there's no sequential relation among these features. Gated GNN [11] was proposed to deal with sequential input, which has a similar mechanism like GRU (gated recurrent unit, Cho et al. (2014), is similar to LSTM). The user interaction history in sequential recommendation system is believed to have a sequential relation that reveals his/her short-term interest, which makes it a suitable scenario for GGNN. Our work contributes in the following aspects:

- We propose the concept of item trend information and investigate different ways to model it. We propose a novel sequential recommender framework: TRec (Trendy Recommender). Three different models are proposed and investigated in TRec to model the item trend representation. Leveraging upon the power of graph neural network, our model is able to aggregate the information from the item's latest activity history. TRec learns the item's trend representation and the item's long-term representation; then, our proposed model combines them together and feeds into the prediction layer alongside with the user's short-term and long-term preferences as well.
- We leverage the power of gated graph neural network on dealing with sequential input, and we incorporate this technique with our model for item trend modeling. Furthermore, a self-attention layer is utilized as a node embeddings learning booster in the item trend representation modeling.
- Comprehensive experiments are conducted in comparison with six state-of-the-art recommender models and different TRec variants as well. Our approach shows promising results over the baseline models. The improvement rate ranges from 4% to 17% in different metrics, when compared to the state-of-the-art models in terms of accuracy on four popular online datasets. Ablation study and hyperparameter evaluation are conducted for different approaches on the item trend representation modeling.

This paper is organized in the following way: Sect. 1 and Sect. 2 outline a brief history of the recommendation system, and then in Sect. 3 we specify the intuitive motivation of our proposed method and briefly explain the basic

algorithm layouts; the detailed discussion will be elaborated in Sect. 4. We evaluate our model with baseline methods and discuss the effect of hyper-parameters in Sect. 5; Ablation study and sensitivity analysis are conducted to assess the different factors that impact on the models. Section 6 summarizes our work and the contribution of this paper and identifies the areas to be explored in future research.

2 Related work

In this section, we will introduce the recent progress of sequential recommendation. After a brief explanation of sequential recommendation, we then introduce the traditional approaches like collaborative filtering based models which play an important role in the early stage of recommendation algorithm. Then, modern deep learning based models like RNN-based or GNN-based algorithms are to be discussed. At last, we show the recent progress on utilizing graph neural network (GNN) on recommendation system algorithm.

2.1 Sequential recommendation

Basically, sequential recommendation system (SRS) is a subset of recommendation systems. Which distinguish SRS from many other recommendation system is its different way of processing the input and generating the output: User's most recent interaction, other than the user's ID itself, is modeled to calculate the most possible next item. Sequential recommendation system is a hot research topic in the research area of recommendation systems for its better user short-term preference representation capacity [7, 12, 13]. The main-stream methods bifurcate into two categories: traditional non-neural approaches and neural-based or deep-learning-based approaches.

2.1.1 Traditional approaches

Most traditional recommender systems are collaborative filtering-based methods. Specifically, they incline to utilize user's history to learn his static preference on the assumption that all user–item interactions in the historical sequence are equally important [7]. In real-world scenario, user behavior is usually not determined merely upon her all-time preference, a basic observation is that the interest of user changes all the time. Earlier sequential recommender approaches employ Markov chain [14] and session-based KNN [15, 16], which fall short in modeling user long-term preferences. Factorization-based methods such as matrix factorization [3] and its variant are widely used in industry for its fast speed and acceptable performance.

2.1.2 Deep learning approaches

In recent days, deep neural networks techniques have been constantly emerging on sequential recommendation systems. RNNs, CNNs and attention mechanism are the mainly adopted approaches for DL-based sequential recommender. RNNs architecture has been well exploited in sequential recommendation domain; Hidasi et al. [8] proposed GRU4Rec which is the first to have introduced RNNs into sequential recommendation; however, GRU4Rec fails to take user information into modeling. Due to potential limitation on sequence length and expensive computing costs, RNNs-based models were less popular as opposed to CNNs and attention-based frameworks. Caser [9] views the embedding matrix of L previous items as an image; therefore, convolution operation could be done for prediction. Yuan et al. [17] proposed NextItNet which utilized residual block CNNs architecture on sequential recommendation tasks. Zhang et al. [6] proposed AttRec which integrated self-attention mechanism into sequential recommendation. Tang et al. [18] discussed sequential model under different temporal contexts and employed a mix of models to cope with different temporal range.

2.2 Graph neural network

GNN [19] can effectively capture node feature from the graph structure. Due to its effectiveness and superior performance in many scenarios, it has captured increasing attention in recommender systems area. In earlier time, inspired by word-embedding-like schemes which exploit skip-gram-like model and achieves great success in natural language processing areas, Perozzi et al. [20] proposed DeepWalk which can effectively learn node representation on graph structures in a random walk fashion. The skip-gram network accepts a node from the random walk as a one-hot vector and maximizes the probability for predicting neighbor nodes. Grover et.al proposed Node2Vec [21] which adopts a similar node embedding scheme. Graph convolution network was first proposed by Kipf et al. [22] which came up with a novel graph convolution operation that is built upon the basis of the dot multiplication of normalized graph Laplacian and graph embedding matrix. It also proved that GCN aggregation is a first-order approximation of spectral graph convolutions. Graph neural network also plays an important part in many areas like community detection [23, 24] and recommender systems [7].

GNN-based SRS Motivated by the recent progress that graph convolution network achieved, researchers of recommender system shift their sight to this new territory. Wu et al. [25] proposed SR-GNN model which constructs

session graph based on item sequences and captures effective transitions of items. Wang et al. [26] proposed neural graph collaborative filtering which adopted graph neural network to collect collaborative signal from user–item graph.

3 Model overview

In this section, we present a brief outline of our proposed model to explain our method in a nutshell. The inputs of our model comprise four parts: the long-term user preference and long-term item representation, the short-term user preference and the item trend information:

- As Fig. 1 shows, before the recommender system makes a recommended item list to a user, the recent interaction trajectory of that user is used as an input to our model for quantifying short-term user preference, which is colored in blue solid line. Here, the interaction of a user can be seen as an explicit or implicit behavior upon an item.
- Then, the item trend information, which is colored in orange dashed line, is learned through the top-k recent users who have interacted with one particular item.
- The long-term user preference and long-term item information are colored in blue dashed line and orange dashed line, respectively. The representations are based upon their index and are further updated through the training phase.

These four inputs are then fed into three different item trend modeling functions to generate the vector representation of item trend. There are three function layers during this process; the first one is the embedding layer which outputs the d-dimensional node embeddings of the inputs. An embedding is a d-dimensional vector which is constructed as a representation for a user or an item. Then, we take advantage of a self-attention layer in order to catch more precisely representation of embedded inputs. A semantic explanation of the modeling process of item trend information can be viewed in a way that the interrelationships of users contribute to the item popularity, which essentially distinguishes our method from other sequential recommender models. At last, the re-expressed embedding matrix of inputs is going to be aggregated through the aggregation layer, and in the prediction layer our model gives a prediction score for each item that user has not interacted. According to the descending ordered list, items with the top-k highest prediction scores are selected as the recommendation result for this particular user.

4 TRec: methodology and model

In this section, we go through each component of our model, as shown in Fig. 1. Firstly, we give a formulated definition on the sequential recommendation, and then we introduce the concept of implicit interactions. In the following, we explain the definition of item trend information and user short-term preference and elaborate how they are modeled through embedding function and self-attention

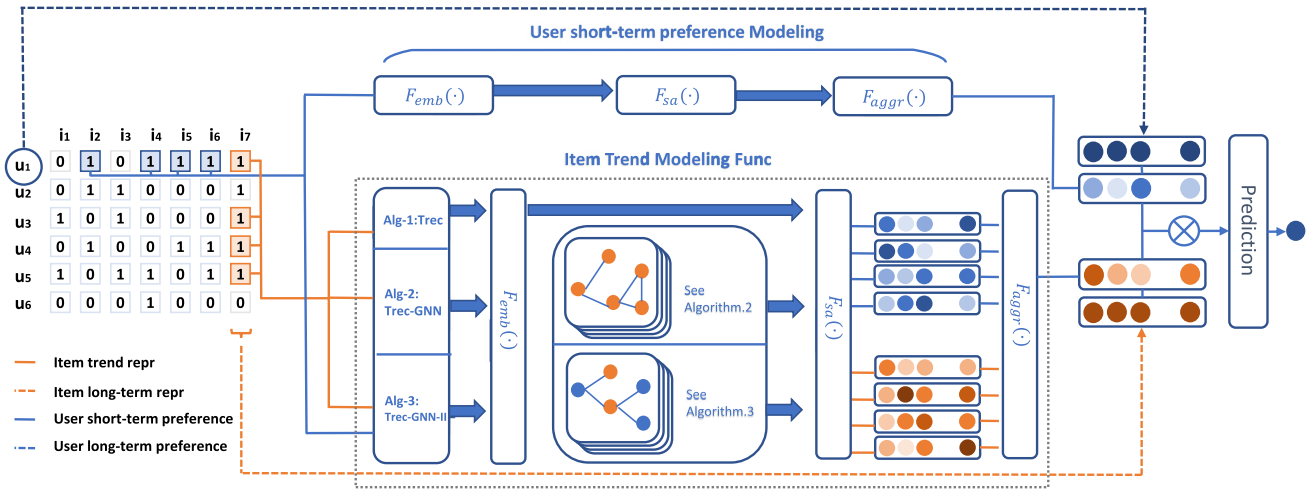

Fig. 1 TRec: Overview of the architecture. The blue and orange squares of the user–item interaction matrix indicate user interaction sequence and top-k recent users, respectively. In this case, top 4 recent users are selected for item trend information modeling. The blue and orange dashed lines modeled from their indices denote user long-term preference and item long-term representation, respectively. The user long-term representation and item long-term representation are gathered together with user short-term representation and item trend information after aggregation step for further prediction jointly

function, and the relationship between those two layers. We also investigate three different variant models which exploit graph neural network to construct better representation for item trend. Next, we demonstrate how these pieces of information together are used jointly to yield a prediction score. At last, we present the time complexity analysis result.

4.1 Formulation of sequential recommendation

At first, to be consistent, we would like to clarify the notation convention used in this paper as follows: Scalars are denoted in plain typeface, a lowercase bold letter denotes a vector, and an uppercase boldface denotes a matrix.

Generally speaking, sequential recommender makes prediction based on a given user's interaction history list. Our method focuses on two aspects of modeling node representation: the first is modeling the user's representation, then the second is modeling the representation for items as well. The representation learning process is finished in the algorithm's training stage. To be more specific, the model learns user's short-term preference from the sequences which are sampled from its interaction history.

A formulated description of our model is given as follows. Assume we have the user–item interaction list of users u which is denoted as $L^u = \{v_k | k \in H_u\}$. Similarly, the item list $L^v = \{u_k | k \in H_v\}$ is the set of users who have interacted with the particular item v, where H_v is the set of item indices that denotes interactions. If our model is regarded as a function $\mathcal{F}(\cdot)$ with all the parameters denoted as Θ, then the sequential recommendation task can be formulated as follows:

$$R = \mathcal{F}[(\phi(L^u), \varphi(\mathbf{L^v})); \Theta], \tag{1}$$

where R is the list of items to be recommended to u, $\mathbf{L^v}$ is the matrix of L^v for all the items in the whole dataset; we use $\phi(\cdot)$ and $\varphi(\cdot)$ to generally refer to any representation learning functions that output vector representations.

Table 1 displays the most frequently used notations in this paper.

4.2 Implicit interaction

Our proposed approach addresses the next item prediction problem with the implicit interaction histories. An implicit interaction is a style of user interaction that does not require the explicit behaviors but still shows the user preference. For example, a click, or a purchase, falls into the category of implicit interaction, while the direct rating behavior of a movie belongs to the explicit interaction. We model the implicit interaction using a two-value system, where 1 indicates the user has interacted with an item and 0 indicates the user dislikes or has not interacted with an item yet. For example, as Table 2 shows, (u_1, v_1) equal to 1 means user u_1 had interacted with item v_1.

Table 1 List of notations

Notation	Meaning
L^u	Interaction history list of u_i
L^v	Set of users have interacted with v_i
H_u	Interacted item indices of u_i
H_v	Indices of users interacted with v_i
L_p^u	Sequence of items u_i recently interacted
L_q^v	Sequence of top-q users interacted with v_i
$\delta^{UsrSP}(u_i)$	Short-term preference of u_i
$\delta^{UsrLP}(u_i)$	Long-term preference of u_i
$\delta^{ITIR}(v_i)$	Item trend information of v_i
$\delta^{ILR}(v_i)$	Long-term representation of v_i
$\mathbf{L_q^v}, \mathbf{L_p^u}$	Embeddings of L_q^v and L_p^u
$\eta^P(u_i, v_j)$	Prediction score of v_j w.r.t u_i
Θ	Parameters to be learned
d	Dimension of embedding
p, q	The lengths of L_p^u and L_q^v
ω, α, β	ω indicates the proportion to which the temporary and long-term user preference contributes,α and β indicates to what extent the item trend information should be taken into account

4.3 User short-term preference modeling

Definition 1 $(UsrSP)$: User short-term preference $(UsrSP)$ which represents the user's recent tendency or fondness on particular items is:

Table 2 An instance of item trend information

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
u_1	1	0	1	1	1	0	1	**1**...(Apr-3-12:30)
u_2	0	0	1	1	0	1	0	0
u_3	1	1	0	1	1	0	0	**1**...(Apr-1-22:23)
u_4	0	0	1	0	1	0	0	**1**...(Feb-2-12:30)
u_5	1	1	0	1	1	0	0	0
u_6	0	0	0	0	1	0	0	**1**...(Feb-1-2:20)
u_7	1	0	0	0	1	0	0	1...(Feb-1-8:32)
u_8	1	1	0	1	1	0	0	1...(Jan-12-2:30)
u_9	0	0	1	0	1	0	0	1...(Jan-11-11:08)
u_{10}	0	0	1	0	1	0	0	1...(Jan-3-4:20)

$$\delta^{UsrSP}(u_i) = \mathcal{F}^{Aggr}(\mathcal{F}^{sa}(\mathcal{F}^{emb}(L_p^u))). \quad (2)$$

The function $Aggr(\cdot), \mathcal{F}^{sa}(\cdot), \mathcal{F}^{emb}(\cdot)$ indicates aggregation layer, self-attention layer and embedding layer, respectively. Here, we first introduce the node embedding function $\mathcal{F}^{emb}(\cdot)$. The embedding vector of each item that the user u has interacted with is obtained by looking up through an item embedding matrix which is initialized with random values and will be updated in the learning process. Thus, the embedding matrix of L_p^u can be denoted by

$$\mathbf{L_p^u} = (\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_p}), \quad (3)$$

where $\mathbf{L_p^u} \in \mathbb{R}^{d \times p}$ and d denotes the dimension of embedding of items.

4.3.1 Node embedding

Let us assume for a interaction history sequence L_u of user u, a recent user–item sequence L_p^u with the length p is extracted for short-term preference modeling, which is represented as

$$L_p^u = \{v_1, v_2, \ldots, v_p\}. \quad (4)$$

Here, we take Table 2 as an example; in this case, we take $p = 4$ for convenience. We can see that the user u_1 has recently watched v_1, v_4, v_5, v_7 those four movies. Then, we have $L_4^{u_1} = \{v_1, v_4, v_5, v_7\}$. We could build the user short-term preference from this L_p^u following the rule of definition 1.

4.3.2 Self-attention layer

The embedded user–item interaction list $\mathbf{L_p^u}$ and item trend information $\mathbf{L_q^v}$ are fed into a self-attention layer $\mathcal{F}^{sa}(\cdot)$ for better representation modeling. Self-attention function [27] is defined as follows:

$$\mathcal{F}^{sa}(\mathbf{V}) = \mathbf{softmax}(\frac{\mathbf{QK^T}}{\sqrt{\mathbf{d}}})\mathbf{V}, \quad (5)$$

where $\mathbf{V}$ is the input matrix of $\mathbf{L_p^u}$ and $\mathbf{Q}, \mathbf{K}$ stand for *query* and *key*, respectively, which are the matrices mapped from input $\mathbf{V}$ with the weight matrices $\mathbf{W_Q}$ and $\mathbf{W_K}$, respectively, where $\mathbf{Q} = ReLu(\mathbf{W_Q V})$, $\mathbf{K} = ReLu(\mathbf{W_K V})$. In addition, the product is fed into a *ReLu* nonlinear activation function. ReLu (Rectified Linear Unit) [28] is a widely used nonlinear activation function in deep learning. It can be formalized as follows:

$$ReLu(x) = max(0, x). \quad (6)$$

On the other hand, $\sqrt{d}$ is a scaling factor used to scale the dot product above, thus avoiding the gradient to end up becoming too small. At last, a softmax layer is used to map the output to a range between 0 and 1, which can be viewed as the affinity of each feature in the matrix. The shape of the output matrix remains the same as input.

4.3.3 Aggregation layer

Aggregation layer $Aggr(\cdot)$ is used to fit the shape of output of embedding matrix in order to be able to perform following dot-product operation. In experiment section we analyzed average aggregation and max aggregation, here we take average aggregation as an example. We aggregate the input matrix, for example, $\mathbf{L_p^u}$, along each row as follows:

$$\delta^{UsrLP}(u_i) = (u_1^+, u_2^+, \ldots, u_d^+). \quad (7)$$

The t th entry of $\delta^{UsrLP}(u_i)$ is aggregated through:

$$u_t^+ = \frac{1}{p}\sum_i \mathbf{L}_{it}^u, \quad (8)$$

where $\delta^{UsrLP}(u_i)$ indicates the short-term preferences of user u_i.

4.4 Item trend information representation modeling

4.4.1 Node embedding

Assuming for the item v, the list of users who have interacted with v is denoted as $L^v = \{u_k | k \in H_v\}$, where H_v is the set of indices that denotes the index of users who have ever interacted with the item v. We extract the latest top q elements of the set L^v for modeling the representation of the trend information of this item. Accordingly, we denote it as

$$L_q^v = (u_1, u_2, \ldots, u_q). \quad (9)$$

Let us take Table 2 as an example. As it shows, the elements in the last column valued 1 indicate that those users have interacted with v_8. Here in the last column, each interaction record is paired with a time-stamp. If we take $q = 4$ in this case, it means that the top 4 recent interactions are used for the item trend information modeling. For convenience, the last column is sorted in the descending order in terms of time-stamp. If we want to select the top 4 records as the latest interaction history, as we boldface in Table 2, we then get the corresponding user list $L_4^{v_8} = (u_1, u_3, u_4, u_6)$.

Since we obtain L_q^v, we build item trend information representation from it as follows:

Definition 2 (*ITIR*): Item Trend Information Representation (*ITIR*) which represents the item's recent trend information is:

$$\delta^{ITIR}(v_j) = \mathcal{F}^{tr}(\mathcal{F}^{emb}(L_p^u)) \tag{10}$$

Where $\mathcal{F}^{tr}(\cdot)$ generally refer to any function that models item trend information representation and outputs a $d-$*dimension* vector.

Similar to what we do in *UsrSP* modeling , $\mathcal{F}^{emb}(\cdot)$ function is implemented by looking up through an embedding function to obtain the embedding matrix of L_q^v:

$$\mathbf{L_q^v} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_q), \tag{11}$$

where $\mathbf{L_q^v} \in \mathbb{R}^{d\times q}$. After we have obtained embedded $\mathbf{L_q^v}$, we then need to implement the $\mathcal{F}_{tr}(\cdot)$ function to model the *ITIR*. In the next section, we will discuss three schemes to implement the function $\mathcal{F}_{tr}(\cdot)$.

4.5 $\mathcal{F}_{tr}(\cdot)$ implementation: three schemes to learn the item trend information

4.5.1 The first implementation of $\mathcal{F}_{tr}(\cdot)$

The first implementation of $\mathcal{F}_{tr}(\cdot)$ is first introduced in our previous paper[10] which TRec is the first time proposed. This scheme could be viewed as a simplified model of the second and the third implementation of $\mathcal{F}_{tr}(\cdot)$ which are going to be elaborated in the forthcoming sections. The basic idea of this implementation comes from the fact that learned node representation tend to have a closer distance for items and users which share similar features in common. Motivated from this idea, the user nodes' representation could be employed to contribute to build item trend representation. In the previous section, we have embedded item's recent user history $\mathbf{L_q^v}$, Here we discuss the first implementation of $\mathcal{F}_{tr}(\cdot)$.

According to the way we build the *UsrSP*, here in the first implementation we adopt a similar way to build *ITIR*. The latest top k item embedding matrix $\mathbf{L_q^v}$ is fed into the self-attention layer $\mathcal{F}^{sa}(\cdot)$ as well.

$$\mathcal{F}^{sa}(\mathbf{L_q^v}) = \mathbf{softmax}(\frac{\mathbf{QK^T}}{\sqrt{\mathbf{d}}})\mathbf{L_q^v}, \tag{12}$$

where $\mathbf{Q}, \mathbf{K}$ stands for *query* and *key*, respectively, which are the matrices mapped from input $\mathbf{L_q^v}$ with the weight matrices $\mathbf{W_Q}$ and $\mathbf{W_K}$, respectively.

Then, the output matrix is aggregated by simply summing the value up along the column. So we could define the first implementation of function $\mathcal{F}_{tr}(\cdot)$ for *ISIR* modeling as follows:

$$\mathcal{F}_{tr}(\mathbf{L_q^v}) = \frac{1}{q}\mathcal{F}^{sa}(\mathbf{L_q^v})^T O \tag{13}$$

where O is $(1, 1, \cdots, 1)^T \in \mathbb{R}^d$.

The algorithm of the first implementation is shown as follows:

Algorithm 1: Algorithm of TRec

Data: $L_p^u \sqsubseteq H^u$
user index i and candidate item index j
Result: $\delta^P(u_i, v_j)$:the possibility of item v_j to be recommended for user u_i

for L_p^u *in All Training Sequences* **do**
  for *item* j *in Target Items* V **do**
    // Generate user short-term representation $\sigma^{UsrSP}(u_i)$
    $\delta^{UsrSP}(u_i) \leftarrow \mathcal{F}^{emb}(L_p^u)$
    $\delta^{UsrSP}(u_i) \leftarrow \mathcal{F}^{sa}(\delta^{UsrSP}(u_i))$
    $\delta^{UsrSP}(u_i) \leftarrow \mathcal{F}^{Aggr}(\delta^{UsrSP}(u_i))$
    // Generate item trend representation $\delta^{ITIR}(v_j)$
    $\delta^{ITIR}(v_j) \leftarrow \mathcal{F}^{tr}(v_j)$
    Predict the score of possibility for next item: $\eta^P(u_i, v_j) = \omega\delta^{UsrLP}(u_i)[\delta^{ILP}(v_j) + \alpha\delta^{ITIR}(v_j)] + (1-\omega)\delta^{UsrSP}[(\delta^{ILP}(v_j) + \beta\delta^{ITIR}(v_j)]$
    $R \leftarrow \eta^P(u_i, v_j)$
  end
  $R \leftarrow TopN(R)$
  // Get Top-N Prediction Results
  Compute Loss and Update
  $\theta \leftarrow \theta + \epsilon\mathcal{F}^{BPR-Loss}(\mathbf{R})$
end
Function $\mathcal{F}^{tr}(v_j)$**:**
  $\delta^{ITIR}(v_j) \leftarrow \mathcal{F}^{emb}(L_q^{v_j})$
  $\delta^{ITIR}(v_j) \leftarrow \mathcal{F}^{sa}(\delta^{ITIR}(v_j))$
  $\delta^{ITIR}(v_j) \leftarrow \mathcal{F}^{Aggr}(\delta^{ITIR}(v_j))$

4.5.2 The second implementation of $\mathcal{F}_{tr}(\cdot)$

We show the way how item trend representation is modeled in previous section; the basic idea is to exploit latent trend information from item's recent users. In the previous section, we adopt a pooling strategy (e.g., mean) which simply aggregates all the user node embeddings within L_q^v to build a vector representation for the each candidate item. We observed that all the user node embeddings of the aggregate pool are independent of each other; as a result, though such vanilla aggregating method do have ability to build item node representation that reflects item trend, there is still room left for improvement in constructing item trend representation.

Here, we discuss the first variant of item trend modeling. We notice an interesting fact that even considering two

items' recent interaction history $L_q^{v_1}$ and $L_q^{v_2}$ of exactly the same group of users, the different chronological order of user's interaction timestamp implies a totally different item trend. For example, considering two sets sorted by chronological order: $L_q^{v_1} = \{u_1, u_1, u_2, u_2\}$ and $L_q^{v_2} = \{u_2, u_2, u_1, u_1\}$, we can see that both of these two sets consist of the same users, assuming that u_1 and u_2 are kung-fu movie lover and romance movie lover, respectively. We could imply that the trend for the first item v_1 is roughly as *action* $\rightarrow$ *romance*, while the trend for the other item v_2 is more like *romance* $\rightarrow$ *action*. However, such sharp distinction among the two items cannot be represented under the method we presented previously.

As a result, we observe that the chronological order of user interaction plays an important role in item trend representation construction. To this end, we come up with an alternative approach to build the item trend representation. The alternative approach firstly builds sequence graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ consisting of nodes from L_q^v. We sort all the elements of L_q^v in chronological order; each element and its next neighbor $(u_i, u_{i+1}) \in \mathcal{E}$ are counted as a directed edge. Figure 3 shows an example of how these user nodes are connected to form a graph. After having built the sequence graph $\mathcal{G}_{v_j}$, we then employ a gated GNN [11] method to update node representation over $\mathcal{G}_{v_j}$. The propagating rule is defined as follows:

$$
\begin{aligned}
\mathbf{h}_{\mathbf{u_i}}^{\mathbf{l}} &= A_{u_i}\mathbf{L}_{\mathbf{q}}^{\mathbf{v}} + \mathbf{b} \\
\mathbf{z}_{\mathbf{u_i}}^{\mathbf{l}} &= \sigma(\mathbf{W_z}\mathbf{h}_{\mathbf{u_i}}^{\mathbf{l}} + \mathbf{U_z}\mathbf{u}_{\mathbf{i}}^{\mathbf{l-1}}) \\
\mathbf{r}_{\mathbf{u_i}}^{\mathbf{l}} &= \sigma(\mathbf{W_r}\mathbf{h}_{\mathbf{u_i}}^{\mathbf{l}} + \mathbf{U_r}\mathbf{u}_{\mathbf{i}}^{\mathbf{l-1}}) \\
\widetilde{\mathbf{u}}_{\mathbf{i}}^{\mathbf{l}} &= tanh(\sigma(\mathbf{W_o}\mathbf{h}_{\mathbf{u_i}}^{\mathbf{l}} + \mathbf{U_o}(\mathbf{r}_{\mathbf{u_i}}^{\mathbf{l}} \odot \mathbf{u}_{\mathbf{i}}^{\mathbf{l-1}})) \\
\mathbf{u}_{\mathbf{i}}^{\mathbf{l}} &= (1 - \mathbf{z}_{\mathbf{u_i}}^{\mathbf{l}}) \odot \mathbf{u}_{\mathbf{i}}^{\mathbf{l-1}} + \mathbf{z}_{\mathbf{u_i}}^{\mathbf{l}} \odot \widetilde{\mathbf{u}}_{\mathbf{i}}^{\mathbf{l}}
\end{aligned}
\tag{14}
$$

where $h_{u_i}^l \in \mathbb{R}^{2|L_q^v|}$ denotes the node vector for u_i which incorporates signals from its neighbors, and l denotes the stage of the node. A_{u_i} is a $\mathbb{R}^{|L_q^v| \times 2|L_q^v|}$ matrix that indicates how nodes communicate with each other. Specifically, matrix A_{u_i} could be seen as two matrices A_{in,u_i} and A_{out,u_i} combined as a whole as Fig. 2 shows, where A_{in,u_i} and A_{out,u_i} indicate the incoming edges and outgoing edges of node u_i, respectively. $z_{u_i}^l$ and $r_{u_i}^l$ are the reset gate and update gate, respectively. $\mathbf{W_z}, \mathbf{U_z}, \mathbf{W_r}, \mathbf{U_r}$ are weight matrices. $\sigma(\cdot)$ and $tanh(\cdot)$ are activation functions. Node information from previous stage $l-1$ is therefore filtered and updated via reset gate and update gate. After l stages, all node embeddings from L_q^v are re-represented with information absorbed from their neighbors.

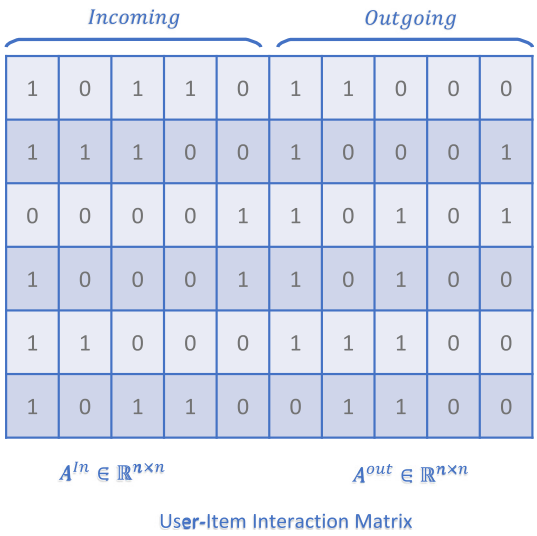

Fig. 2 Incoming matrix $\mathbf{A}^{in}$ and outgoing matrix $\mathbf{A}^{out}$ of item trend graph are concatenated together

Algorithm 2: Algorithm of Variant I:TRec-GNN

Data: $L_p^u \sqsubseteq H^u$
user index i and candidate item index j
Result: $\delta^P(u_i, v_j)$:the possibility of item v_j to be recommended for user u_i

Function $\mathcal{F}_{var}^{tr}(v_j)$ **:**
  // Build Item Trend Graph $\mathcal{G}_{v_j}$
  For node set $\mathcal{V}$ from $\mathcal{G}_{v_j} = \{\mathcal{V}, \mathcal{E}\}$, initialize the node hidden states at stage zero.
  Update node embeddings of graph $\mathcal{G}_{v_j}$ under propagation rule of equation (14)
  $\mathbf{L}_{\mathbf{q}}^{\mathbf{v}} = (\mathbf{h}_{\mathbf{u_1}}^{\mathbf{l}}, \mathbf{h}_{\mathbf{u_2}}^{\mathbf{l}}, \cdots, \mathbf{h}_{\mathbf{u_q}}^{\mathbf{l}})$
  $\mathbf{L}_{\mathbf{q}}^{\mathbf{v}} \leftarrow \mathcal{F}^{sa}(\mathbf{L}_{\mathbf{q}}^{\mathbf{v}})$
  $\delta^{ITIR}(v_j) \leftarrow \mathcal{F}^{Aggr}(\mathbf{L}_{\mathbf{q}}^{\mathbf{v}})$

We denote this output of $l-th$ layered GNN as $\mathbf{L}_{\mathbf{q}}^{\mathbf{v}} = (\mathbf{u}_{\mathbf{1}}^{\mathbf{l}}, \mathbf{u}_{\mathbf{2}}^{\mathbf{l}}, \cdots, \mathbf{u}_{\mathbf{q}}^{\mathbf{l}})$. This node embeddings set is served as a kind of representation of item trend; similar to what we do in vanilla item trend modeling which is discussed previously, we could stack self-attention layer and aggregation layer on it as well.

4.5.3 The third implementation of $\mathcal{F}_{tr}(\cdot)$

In Variant I, we discuss an alternative item trend modeling method with GNN, The item trend graph we used is established on chronological ordered user interaction records. It reveals the importance of modeling the item trend representation under a temporal perspective. In this fashion, we obtain better item node representations. We observe that the item trend graph $\mathcal{G}_{v_j}$ initializes all node embeddings at hidden state zero $\mathbf{h}_{\mathbf{u}}^{\mathbf{0}}$ with Gaussian randomized vectors. In other words, the node embedding of an

element u_i only represents the corresponding user's long-term preferences.

A natural idea is to improve the user node embeddings with extra information from user's short-term preference representations. In previous sections, we have already discussed the way to construct user's short-term preference representation $\delta^{UsrSP}(u_i)$, so we could conveniently incorporate this vector with its long-term preference representation. And by combining the short-term and long-term preference together, we have a refined vector which is to be served as the initial hidden state of each node $\mathbf{h_u^0} \in \mathcal{G}_{\mathbf{v_j}}$. This can be achieved as follows:

$$\begin{aligned}\mathcal{V} =&\{\mathbf{h_u^0} = \omega\delta^{UsrLP}(u_i) \\ &+ (1-\omega)\delta^{UsrSP}(u_i)|\mathbf{h_u^0} \in \mathcal{G}_{v_j}\}\end{aligned} \tag{15}$$

where $\mathcal{V} \in \{\mathcal{G} = (\mathcal{V}, \mathcal{E})\}$ indicates the node set.

Algorithm 3: Algorithm of Variant II:TRec-GNN-II

Data: $L_p^u \sqsubseteq H^u$
user index i and candidate item index j
Result: $\delta^P(u_i, v_j)$:the possibility of item v_j to be recommended for user u_i

Function $\mathcal{F}_{var_{II}}^{tr}(v_j)$ **:**
  // Build Item Trend Graph $\mathcal{G}_{v_j}$
  For node set $\mathcal{V}$ from $\mathcal{G}_{v_j} = \{\mathcal{V}, \mathcal{E}\}$:
  $\mathcal{V} = \{\mathbf{h}_{\mathbf{u_i}}^{\mathbf{0}} = \omega\delta^{UsrLP}(u_i) + (1-\omega)\delta^{UsrSP}(u_i)|\mathbf{h_{u_0}} \in \mathcal{G}_{v_j}\}$
  Update node embeddings of graph $\mathcal{G}_{v_j}$ according to equation (14)
  $\mathbf{L_q^v} = (\mathbf{h_{u_1}^l}, \mathbf{h_{u_2}^l}, \cdots, \mathbf{h_{u_q}^l})$
  $\mathbf{L_q^v} \leftarrow \mathcal{F}^{sa}(\mathbf{L_q^v})$
  $\delta^{ITIR}(v_j) \leftarrow \mathcal{F}^{Aggr}(\mathbf{L_q^v})$

4.5.4 Long-term representation for users and items

We denote the long-term representation of the user u as $\mathbf{u}_i$. Accordingly, the long-term representation of the item v is denoted as $\mathbf{v}_i$. In our approach, the long-term representations for users and items are two low-rank vectors with d dimensions, which are obtained through a look-up operation from separated embedding matrices. Here, d is a hyperparameter which is specified by the user.

Definition 3 (*UsrLP*): User Long-term Preference (UsrLP) is defined as:

$$\delta^{uLP}(u_i) = \mathcal{F}^{EMB}(u_i) \tag{16}$$

where u_i is the index of the ith user and $\mathcal{F}^{EMB}(\cdot)$ is the node embedding function which receives u_i as input and outputs a d-dimension vector as the user's long-term preference.

Definition 4 (*ILP*): Item Long-term Representation (ILR) is defined as:

$$\delta^{ILP}(v_i) = \mathcal{F}^{EMB}(v_i) \tag{17}$$

where v_i is the index of the ith item and $\mathcal{F}^{EMB}(\cdot)$ is the node embedding function which receives v_i as input and outputs a d-dimension vector as the item's long-term representation.

For example, the index i of one particular user u_i is used to get a corresponding embedding vector $\mathbf{u}_i$. The parameters of the embedding are updated via the back-propagating process in each training epoch.

4.6 Prediction layer

The prediction layer aims to generate a scalar value for each candidate item which implies that the possibility of this candidate item becomes the user's next interaction object. In prediction layer, we exploit user's long-term and short-term preference information: *UsrSP* and *UsrLP*, and item trend information representation *ITIR* and its long-term representation *ILR* to calculate the output.

Definition 5 We define the prediction function as follows:

$$\begin{aligned}\eta^P(u_i, v_j) =&\omega\delta^{UsrLP}(u_i)[\delta^{ILP}(v_j) + \alpha\delta^{ITIR}(v_j)]+ \\ &(1-\omega)\delta^{UsrSP}[(\delta^{ILP}(v_j) + \beta\delta^{ITIR}(v_j)]\end{aligned} \tag{18}$$

where $\eta^P(u_i, v_j)$ is the possibility of next candidate item to be recommended by our recommender algorithm, j is the index of the candidate item, and i is the index of the current user to be recommended. ω indicates the proportion to which the short-term and long-term user preferences contribute, α and β specify to what extent the item trend information should be taken into account. When $\omega = 1, \alpha = 0$, it degrades to a matrix factorization-based model [3].

4.6.1 Loss function definition

Inspired by the BPR-Opt proposed by Rendal et.al [29], which has been proved to provide a better ranking quality than the rating prediction-based optimization methods in implicit interaction scenarios, the loss function in our approach is defined as:

$$\sum_{(u,i,j)\in D_s} \ln\sigma[\eta^P(u_i, v_j) - \eta^P(u_i, v_j)] - \lambda_\Theta||\Theta||^2, \tag{19}$$

where $D_s = \{(u, i, j|i \in L^u, j \in I\backslash L^u)\}$ and I denotes the items set. The semantic explanation of D_s is that user u is assumed to prefer i over j. $\eta^P(u_i, v_j)$ is the prediction score of the user u on the item i and j, and $\lambda_\Theta||\Theta||^2$ is the regularization term.

Let's take the movie recommendation as an example; our loss function aims to minimize the error so that a user

has a greater possibility to watch the movie i over the movie j.

4.7 Time complexity analysis

4.7.1 Time complexity of $\mathcal{F}^{tr}(\cdot)$

In the first implementation of $\mathcal{F}^{tr}(\cdot)$, its time complexity mainly depends on the time cost of the self-attention function $\mathcal{F}^{sa}(\cdot)$. Assume the single-user interaction sequence to be L_q^v, and L_p^u to be the sequence length of users who have interacted with one item recently. The time complexity is $\mathcal{O}(|L_q^v|^2 d + |L_p^u|^2 d)$. Thanks to the parallelizable nature of self-attention mechanism, our method has a relatively low sequential complexity of $\mathcal{O}(1)$, compared with the RNN-based method such as GRU4Rec [8] which has a sequential complexity of $\mathcal{O}(n)$, since it has to wait for the output of time step $t-1$.

4.7.2 Time complexity of $\mathcal{F}^{tr}_{Var}(\cdot)$

Since graph neural network is used to update the node embeddings, the time complexity is $\mathcal{O}(md)$, where m indicates the count of edges in the graph. Since we build graph from nodes of $L_q^{v_j}$, the number of edges equals $|L_q^{v_j}| - 1$.

4.7.3 Time complexity of $\mathcal{F}^{tr}_{Var_{II}}(\cdot)$

In the third implementation of $\mathcal{F}^{tr}(\cdot)$, we adopt an alternative way to initialize the node embedding values of graph $\mathcal{G}_{v_j}$; for each node u_i of $\mathcal{G}_{v_j}$, the time complexity for building $\delta^{UsrSP}(u_i)$ is $|L_p^{u_i}|$. As for every $\mathcal{G}_{v_j}$, the node number equals $|L_q^{v_j}|$, so we have the time complexity of $\mathcal{F}^{tr}_{var_{II}}(\cdot)$ equals $\mathcal{O}(|L_p^{u_i}||L_q^{v_j}|d + |L_q^{v_j}|d)$.

The empirical study results evidence that our method runs the order of magnitude faster than other RNN-based and CNN-based methods.

5 Experiments

We proposed the concept of item trend representation and methods to generate it; we need to confirm and validate the effectiveness of our models. In this section, extensive experiments are conducted to verify the effectiveness, sensitivity and efficiency of our proposed model. Accordingly, we would like to answer the following research questions:

- RQ1: Does our approach outperform the state-of-the-art models?
- RQ2: We proposed the concept of item trend representation, is it useful in the sequential recommendation tasks? If so, how do the key hyperparameters affect the performance of our approach?
- RQ3: We utilize graph neural network to generate item trend representation, to what extent can such model impact on and influence the performance?

5.1 Experimental settings

5.1.1 Dataset

We perform our experiments with four public datasets: Three of them are the subsets of Amazon Custom Review Dataset: Luxury, Software and Digital. Amazon Custom Review Dataset [30] is a widely accepted stable benchmark dataset for recommendation systems. The rest one dataset we adopt is MovieLens100K[1], which is an online movie Web site for movie recommendation and building custom movie taste profiles. The statistics of these four datasets are demonstrated in Table 3, where the median in the header row means the median of rating counts per user. Based on this statistics, we conclude the facts as follows:

- MovieLens100K has the largest median of ratings per user (70.5 per user), which implies that the users from movie Web sites behave more actively than those from the e-commerce shopping Web sites. Luxury subset comes a close second with 32 ratings per user in terms of median.
- Software and digital subsets have a significantly fewer ratings per user. It suggests that users tend to buy fewer items of those categories.

5.1.2 Evaluation measures

The dataset is split into three portions: 70 % of original dataset is used for training and the rest is split into two parts: 20 % for validation and 10 % for testing. We use recall and NDCG (normal discounted cumulative gain) to evaluate our approach versus other models. Recall@K indicates the ratio of users that have interacted with the items over top-K recommended items, given by our prediction algorithm, while NDCG takes the position into account, which is often used in measuring the effectiveness of Web search engine algorithms. NDCG measures the usefulness or gain of a document based on its position in the result list. The gain is accumulated from the top of the

[1] https://grouplens.org/datasets/MovieLens/.

Table 3 Dataset statistics

Dataset	Users	Items	Total rating counts	Median
Luxury	12,369	416,425	536,554	32
ML100K	610	9,724	100,000	70.5
Software	21,663	375,147	459,436	3
Digital	32,589	324,040	371,344	2

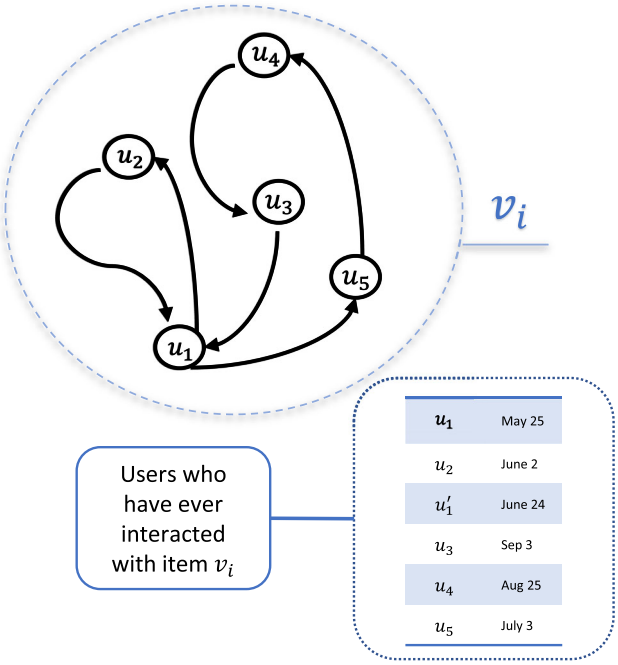

Fig. 3 Item trend graph construction of item node v: User nodes of item's recent records are connected with edges which indicate a chronological order between them

result list to the bottom, with the gain of each result discounted at lower ranks.

5.1.3 Baseline methods

The following competitive methods which are popular in the sequential recommendation tasks are compared with our proposed approach.

- *BPR-MF* BPR-MF [29] is a well-known latent factor model for recommendation system, where rating is calculated through a dot product of two low-rank user and item latent vectors.
- *FPMC* FPMC (factorization machines) [31] FPMC is a representative baseline for next-basket recommendation, which integrates the MF with first-order MCs.
- *GRU4Rec* GRU4Rec [8] is the first model that applies RNN to the sequential recommendation and does not consider a user's identity. The input of GRU4Rec is a set of items, and the embedded items matrix is fed into the stacked GRU layers for next item prediction.
- *AttRec* AttRec [6] utilizes the self-attention mechanism from NLP to infer the item–item relationship from the user historical interactions. It also takes user's transient interest into consideration.
- *Caser* Caser [9] is a convolution-based sequential recommendation model. It captures the high-order Markov chains via applying the convolution operations on the recent user interaction sequences.
- *HGN* HGN (hierarchical gating networks for sequential recommendation) [32] adopts a hierarchical gating architecture to select what item features can be passed to the downstream layers from the feature and instance level. HGN outperforms several recent state-of-the-art models on different datasets.

5.2 RQ1: Performance comparison

The performance comparison result is shown in Fig. 4. We compare the three item trend representation modeling approaches with all other six baseline recommendation models. It is demonstrated that our proposed three item trend information integrated models consistently achieve better performance over all other baseline models. The superior performance confirms our hypothesis that item trend information helps boost the recommender performance; the experiment result gives a convincing positive answer to RQ1. Based on the results, we have two observations as follows:

- By integrating the item trend information and the user short-term preference together with the long-term user and item representation, our model has significantly enhanced the effectiveness of sequential recommendation tasks. This table also obviously shows the listed non-neural approaches (i.e., MF and FPMC), which are significantly outperformed by our method with an up to 17 percent gain. An intuitive explanation is that these latent factor-based models view users and items as being static. Hence, they neglect the potential and latent information within them, such as the item trend information and the user short-term preference.
- Compared to the latest neural-based methods, our model achieves an encouraging gain in performance. We can see that our proposed model exhibits the improvements over other baseline models from 4% to 17% on different datasets. It is noteworthy that our model achieves a more competitive gain on the Luxury dataset. A reasonable explanation is that the luxury item goes out of fashion in a relatively shorter time span.
- The GNN-powered models of TRec achieve better performance over the vanilla TRec model on most tests.

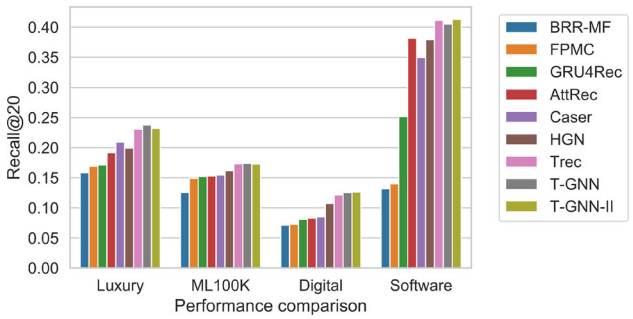

Fig. 4 Performance comparison over other baseline models

It proves that with GNN employed in the item trend representation building phase, the generated item trend representation could obtain a better accuracy. We also notice that TRec-GNN-II which takes advantage of higher-level neighbors of items on node embedding initialization surpasses the other two models on several tests. These results suggest that there is an association between a more sophisticated item trend representation modeling and a better performance on recommendation.

5.3 RQ2: Hyperparameter and ablation analysis

In order to answer RQ2, we design a series of experiments aiming to figure out the effect of the crucial hyperparameters which will affect the performance of our model. We do the ablation analysis as well which aims to investigate the effectiveness of the item trend information. Item trend information plays a crucial role in our approach and distinguishes our method from other sequential recommendation methods. The experimental works presented here will provide one of the first investigations to assess the impact of item trend information on sequential recommendation system. We will discuss all three models for item trend representation modeling and influence of important hyperparameters with separated paragraphs.

5.3.1 Evaluation for TRec

We assess the factors that influence the performance of the first item trend representation model: The first one is the sequence length in modeling, and the other factor is the proportional coefficient which determines to what portion the item trend information is absorbed with the item long-term representation.

The hyperparameter q directly influence the first factor, and the hyperparameter ω, β andα significantly affect upon the second factor. We study the impact of these two factors as follows.

5.3.2 Window length q for trend modeling

The sequence length q of the recent user defines the time span of the trend for an item. Based on the result shown in Fig. 7, we have the following observations:

- The performance goes slightly better when the length parameter q is set to a higher value.
- The optimal value of q varies and depends on the dataset we choose. The datasets in which items have longer user interaction history require a relatively larger value of q; therefore, the item trend is well represented. For example, a mechanical keyboard has gone out of fashion since 2000, but remains popular in the niche market. As such, one or two recent purchase records cannot represent its falling trend.

5.3.3 Sensitivity analysis: ω,α and β

The proportion parameters determine how much percentage the item trend information should be considered, and thus affect the final prediction score. To be more concretely, ω indicates the proportion of the long-term user preference, and correspondingly $1-\omega$ reflects the proportion of the short-term user preference. Similarly, α and β are the proportions of the item trend information.

The impact of proportion parameters is shown in Fig. 5. The optimal value of these parameters varies when different datasets are adopted. As we can obviously see from this figure, the higher values of α and β result in the better performance on the four datasets.

We can also find that the optimal value of ω varies on four datasets; however, the performance downgrades rapidly as ω goes toward 1. This reveals the importance of user short-term preference.

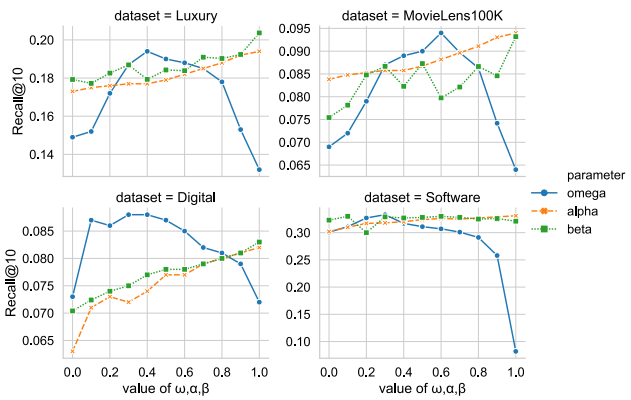

Fig. 5 Effects of the hyperparameters: ω, α, β on luxury and MovieLens datasets

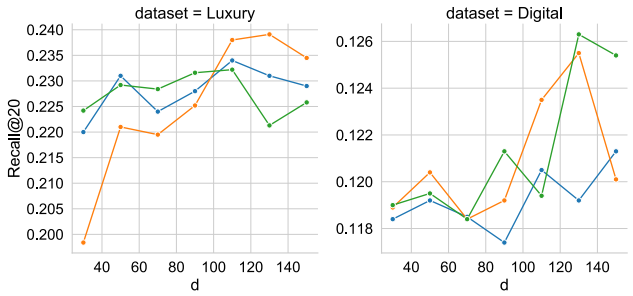

Fig. 6 Effects of dimension of embedding: d

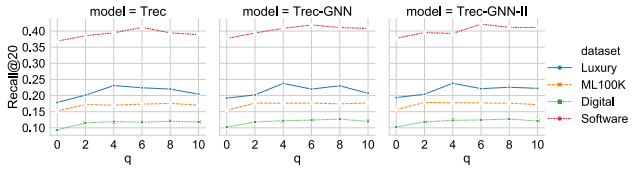

Fig. 7 Effect of q for three ITR models

5.3.4 Dimension of embedding

Embedding vector carries the information of users and items. The dimension of embedding vector dictates the capacity of information of a user or an item. We test the effect of varying dimension lengths on two datasets: Amazon-Luxury and Software. We present the effect of dimension of embedding d on Recall@10 in Fig. 6. It shows that embedding with a dimension less than 40 is incapable to represent enough information of a user or an item. As d grows larger, the performance goes up with an unstable fluctuation. Generally speaking, the performance of all the models behaves much better when d is larger than 100.

5.3.5 Evaluation for TRec-GNN

As was mentioned in the previous chapter, the TRec-GNN variant models item trend information representation (ITIR) in a different way that employs GNN to learn the ITIR from the item's recent interaction history. We seek to investigate the effect of the window length q of L_q^v via adjusting the parameter value of q over four datasets. As Fig. 7 shows, we have conclusions observed as follows:

- Proper window length q boosts the performance of the model; on all four datasets, the best performance is achieved when an optimal value of q is adopted.
- Longer window length does not always come with better performance; as the window length grows larger and becomes closer to the item's total user interaction count $|H_v|$, the performance degrades to the similar level of BPR-MF-based model.
- The optimal value of window length q varies with the dataset adopted. We suggest that it is related to the item's average interaction count; as for a longer interaction record of an item, the trend effect becomes more obvious and vice versa.

5.3.6 Evaluation for TRec-GNN-II

Compared to TRec, in which the item's recently interacted user's long-term representation $\delta^{UsrLP}(u_i)$ is used for item trend modeling, TRec-GNN-II further incorporates user's short-term and long-term representation together for building the GNN of G_{v_i}. Hence, a properly adjusted user sequence length affects the representation learning ability of GNN. We evaluate the effect of user sequence length p by conducting experiment on four datasets with a group of different p value settings. Based on the results from Table 5, we have observation as follows:

- The performance of TRec-GNN-II is boosted when user sequence length is set larger than 0; we suggest that it is benefited from incorporating user's long-term and short-term representations into the item trend graph G_{v_i}'s node embedding initialization.
- The optimal value of p varies with different datasets, and the performance fluctuation is not obvious. We suggest this might be attributed to the GNN propagation mechanism.

5.3.7 Ablation analysis

In order to answer RQ2 and find out the effectiveness of different components in our proposed framework, to this end, we take apart our model and evaluate the performance with part of its component taken away. The results are based on Amazon-Luxury and MovieLens100K. Recall@10 and NDCG@10 are used as the evaluating metrics for our test. R@10 in Table IV stands for Recall@10. The results shown in Table 6 demonstrate the effect of these different model variations. The default setting achieves the ideal marks over the Luxury and ML100K two datasets. We can see our model with the self-attention layer taken away decreased in performance. The reason might be that the self-attention layer efficiently captures the interrelation of the variables within the features of user and item. We also notice the impact of removing item trend information comes with a decrease in performance. The Amazon-Luxury dataset has a larger rate of performance decreasing, which is supposed to attribute to a shorter time span of the item trend. Equation (5) shows that the user short-term preference plays an important role in both datasets. Removing the user short-term

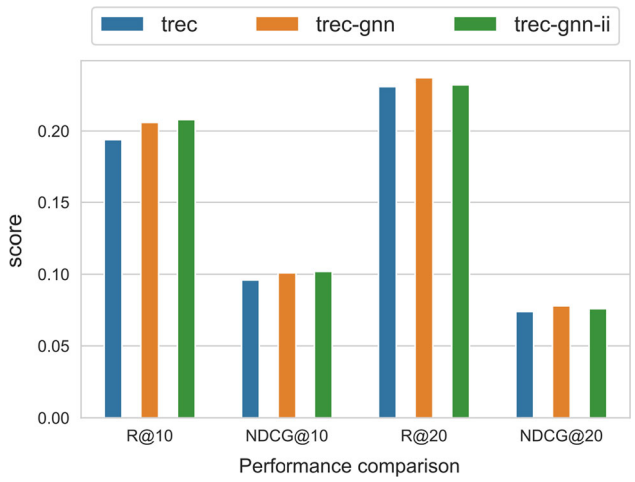

Fig. 8 Comparison among three item trend representation models

representation leads to a consistent performance degradation in our test. Equation (6) shows a significant loss in performance. Compared to Eq. (5) which suffers a relatively modest decrease, it seems that the item trend information makes a compensation on the accuracy in some way. We also experiment with different attentive matrix aggregation methods. In the default situation, the average method is adopted. We test the max aggregation as an alternative way. The result in (7–8) shows a slight fluctuation in performance. So we conclude that the impact of aggregation is related to different datasets.

5.4 Comparison among different item trend representations

We compare performance among the three different item trend representation variant methods (see Sect. 4) over Amazon-Luxury dataset. We replace the item trend representation construction method of TRec with different variant methods. Each variant method is tested for five turns, and we report the average metrics for them.

Figure 8 displays that GNN-based variant indeed improves the model's ability of item trend representation. Both Variant-I and Variant-II outperform the vanilla TRec model in terms of Recall@20 and NDCG@20. We see no remarkable disparity between Variant-I and Variant-II; one possible reason is that the initial hidden states of nodes within item trend graph do not count for much; however, introduction of GNN-based representation is what contributes the most for these variant methods.

Table 4 Training time cost comparison (per epoch) in terms of seconds

Dataset	GRU4Rec	Caser	AttRec	HGN	TRec
ML100k	16	14	**1.8**	2.5	1.9
Luxury	40	32	7.8	9.0	**7.8**
Digital	30	23	**5.7**	6.5	5.8
Software	33	28	**6.5**	7.3	6.6

Table 5 Effect of p for TRec-GNN-II

p	Luxury	ML100k	Digital	Software
0	0.2105	0.1624	0.1101	0.3756
2	0.2252	0.1689	0.1187	0.3845
4	**0.2398**	0.1727	0.1201	**0.4102**
6	0.2325	**0.1743**	**0.1225**	0.4061
8	0.2352	0.1722	0.1218	0.3917
10	0.2341	0.1727	0.1212	0.3964

Table 6 Ablation analysis, w/ and w/o stand for with and without, iti stands for item trend information

Architecture	Amazon-Luxury		ML100k	
	R@10	NDCG@10	R@10	NDCG@10
(1) TRec	**0.194**	**0.096**	**0.103**	**0.096**
(2) w/o Self-Att	0.182	0.085	0.095	0.083
(3) w/o ITI	0.174	0.078	0.093	0.079
(4) w/o Self-Att&ITI	0.170	0.074	0.089	0.075
(5) w/o u+	0.179	0.081	0.094	0.081
(6) w/o u+&ITI	0.132	0.051	0.064	0.051
(7) aggr-avg	0.194	0.096	0.103	0.096
(8) aggr-max	0.192	0.094	0.103	0.095

5.5 Training efficiency

We conduct the training efficiency test over four different benchmark datasets. Factorization-based models are not tested in this comparison as their recommendation accuracy is outperformed by the state-of-the-art models. All the tests are running over 100 epochs and the average time cost per epoch is recorded for evaluation. Table 4 shows that the deep learning-based models like GRU4Rec [8] and Caser [9] cost a huge amount of time for training each epoch. Our model TRec achieves a significantly lower runtime cost over other neural-based baseline models. Therefore, our proposed method has a promising potential to be employed into the real industrial practice.

6 Conclusion and future work

This paper has discussed the concept of item trend information and methods to model it on sequential recommendation systems. In this work, we propose a novel sequential recommendation model which extracts information from user–item interaction history to construct item trend information representation. We design three different approaches which aim to construct the item trend representation. The second aim of this study is to investigate to what extent the performance of the recommender is affected and influenced by the new proposed approach. The main contribution of this work is that we incorporate the item trend information representation into the next item prediction task. We leverage the power of gated graph neural network to gather information from nodes which are sampled from candidate item's interaction history. We carry out a series of experiments to evaluate the performance of our model and the effect of the hyperparameters. The results demonstrate that our proposed model achieves far better performance as opposed to the state-of-the-art models.

In this paper, we have proved that the graph neural network-based model offers significant advantage in generating item trend information representations. We still observe that the learned user and item representations are based on the current existing database; therefore, it still remains a challenge task for cold-start problems. Thus, we will seek different ways to generate user and item node embeddings efficiently. In recent future, we will be interested to explore self-supervised learning and its potential to be integrated into item trend information representation learning.

Acknowledgements This work is supported in part by Griffith Industry Collaborative Grant. This work is supported in part by the National Key Research and Development Program of China under Grant 2017YFE0117500. This work is supported in part by the Natural Science Foundation of the Higher Education Institutions of Jiangsu Province (Grant No. 17KJB520028), Tongda College of Nanjing University of Posts and Telecommunications (Grant No. XK203XZ18002).

Compliance with ethical standards

Conflict of interest We declare that we have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

1. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, WWW 2001. Association for Computing Machinery Inc., Apr 2001, pp 285–295
2. Pazzani M J, Billsus D (2007) Content-based recommendation systems. In: Lecture notes in computer science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol 4321 LNCS, 2007, pp 325–341
3. Hayakawa M (2015) MF Tech. Earthquake prediction with radio techniques 2015:199–207
4. Jannach D, Lerche L, Jugovac M (2015) Aaptation and evaluation of recommendations for short-term shopping goals. In: RecSys 2015—Proceedings of the 9th ACM conference on recommender systems. Association for Computing Machinery Inc, Sep 2015, pp 211–218
5. Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv 52(1):2019
6. Zhang S, Tay Y, Yao L, Sun A 2018) Next item recommendation with self-attention. Available: http://arxiv.org/abs/1808.06414 (online)
7. Fang H, Danning Z, Yiheng S (2019) Deep learning for sequential recommendation: algorithms, influential factors, and evaluations. In: SIGIR 2019—Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval. Association for Computing Machinery Inc, 2019, pp 1281–1284
8. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2016) Session-based recommendations with recurrent neural networks. In: 4th International conference on learning representations, ICLR 2016—conference track proceedings, 2016. Available: http://arxiv.org/abs/1511.06939 (online)
9. Tang J, Wang K (2018) Personalized top-N sequential recommendation via convolutional sequence embedding. In: WSDM 2018—Proceedings of the 11th ACM international conference on web search and data mining, vol 2018, pp 565–573
10. Tao Y, Wang C, Yao L, Li W, Yu Y (2020) TRec: Sequential recommender based on latent item trend information. In: Proceedings of the international joint conference on neural networks
11. Li Y, Zemel R, Brockschmidt M, Tarlow D (2016) Gated graph sequence neural networks. In: 4th International conference on learning representations, ICLR 2016—Conference Track Proceedings, 2016
12. Song X, Li J, Sun SJ, Yin H, Dawson P, Doss R (2020) SEPN: a sequential engagement based academic performance prediction model. IEEE Intell Syst
13. Li J, Cai T, Deng K, Wang X, Sellis T, Xia F (2020) Community-diversified influence maximization in social networks. Inf Syst 92:101522
14. Davidson J, Liebald B, Liu J, Nandy P, Van Vleet T (2010) The YouTube video recommendation system. In: RecSys'10—Proceedings of the 4th ACM Conference on Recommender Systems, 2010, pp 293–296 (Online). Available: www.youtube.com/videos
15. Jannach D, Ludewig M (2017) When recurrent neural networks meet the neighborhood for session-based recommendation. RecSys 2017—Proceedings of the 11th ACM conference on recommender systems, 2017, pp 306–310 (Online). Available: http://dx.doi.org/10.1145/3109859.3109872
16. He R, McAuley J (2017) Fusing similarity models with markov chains for sparse sequential recommendation. In: Proceedings—IEEE international conference on data mining, ICDM, 2017, pp 191–200 (Online). Available: https://sites.google.com/a/

17. Yuan F, Karatzoglou A, Arapakis I, Jose J M, He X (2019) A simple convolutional generative network for next item recommendation. In: WSDM 2019—Proceedings of the 12th ACM international conference on web search and data mining, vol 19, 2019, pp 582–590 (Online). Available: https://doi.org/10.1145/3289600.3290975
18. Tang J, Belletti F, Jain S, Chen M, Beutel A, Xu C, Chi EH (2019) Towards neural mixture recommender for long range dependent user sequences. In: The web conference 2019—proceedings of the world wide web conference, WWW 2019, pp 1782–1793 (Online). Available: http://arxiv.org/abs/1902.08588, http://dx.doi.org/10.1145/3308558.3313650
19. Bacciu D, Errica F, Micheli A, Podda M (2019) A gentle introduction to deep learning for graphs. Technical Report, 2019 (Online). Available: http://arxiv.org/abs/1912.12693
20. Perozzi B, Al-Rfou R, Skiena S (2014) DeepWalk: online learning of social representations. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710 (Online). Available: http://arxiv.org/abs/1403.6652, http://dx.doi.org/10.1145/2623330.2623732
21. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, vol 13–17, 2016, pp 855–864 (Online). Available: http://arxiv.org/abs/1607.00653
22. Kipf T N, Welling M (2016) Semi-supervised classification with graph convolutional networks, pp 1–14, 2016 (Online). Available: http://arxiv.org/abs/1609.02907
23. Liu F, Xue S, Wu J, Zhou C, Hu W, Paris C, Nepal S, Yang J, Yu PS (2020) Deep learning for community detection: progress, challenges and opportunities. Technical Report
24. Jia J, Segarra S, Schaub MT, Benson AR (2019) Graph-based semi-supervised and active learning for edge flows. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 761–771, 2019 (Online). Available: https://doi.org/10.1145/3292500.3330872
25. Wu S, Tang Y, Zhu Y, Wang L, Xie X, Tan T (2018) Session-based recommendation with graph neural networks (Online). Available: http://arxiv.org/abs/1811.00855%0A, http://dx.doi.org/10.1609/aaai.v33i01.3301346
26. Wang X, Zhu W, Liu C (2019) Social recommendation with optimal limited attention. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 1518–1527, (Online). Available: https://doi.org/10.1145/3292500.3330939
27. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 17:5999–6009
28. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: ICML 2010—Proceedings, 27th international conference on machine learning, 2010, pp 807–814
29. 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, UAI 2009, pp 452–461
30. Ni J, Li J, McAuley J (2019) Justifying recommendations using distantly-labeled reviews and fine-grained aspects. arXiv, pp 188–197
31. Rendle S (2010) Factorization machines. In: Proceedings—IEEE International Conference on Data Mining, ICDM, 2010, pp 995–1000 (Online). Available: http://www.libfm.org
32. Ma C, Kang P, Liu X (2019) Hierarchical gating networks for sequential recommendation. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 825–833 (Online). Available: https://doi.org/10.1145/3292500.3330984