Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer

Ziwei Fan*, Zhiwei Liu*
Department of Computer Science, University of Illinois at Chicago
USA
{zfan20,zliu213}@uic.edu

Jiawei Zhang
IFM Lab, Department of Computer Science, University of California, Davis
USA
jiawei@ifmlab.org

Yun Xiong
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
China
yunx@fudan.edu.cn

Lei Zheng
Pinterest Inc.
USA
lzheng@pinterest.com

Philip S. Yu
Department of Computer Science, University of Illinois at Chicago
USA
psyu@uic.edu

ABSTRACT

In order to model the evolution of user preference, we should learn user/item embeddings based on time-ordered item purchasing sequences, which is defined as Sequential Recommendation (SR) problem. Existing methods leverage sequential patterns to model item transitions. However, most of them ignore crucial temporal collaborative signals, which are latent in evolving user-item interactions and coexist with sequential patterns. Therefore, we propose to unify sequential patterns and temporal collaborative signals to improve the quality of recommendation, which is rather challenging. Firstly, it is hard to simultaneously encode sequential patterns and collaborative signals. Secondly, it is non-trivial to express the temporal effects of collaborative signals.

Hence, we design a new framework **T**emporal **G**raph **S**equential **Rec**ommender (TGSRec) upon our defined continuous-time bipartite graph. We propose a novel Temporal Collaborative Transformer (TCT) layer in TGSRec, which advances the self-attention mechanism by adopting a novel collaborative attention. TCT layer can simultaneously capture collaborative signals from both users and items, as well as considering temporal dynamics inside sequential patterns. We propagate the information learned from TCT layer over the temporal graph to unify sequential patterns and temporal collaborative signals. Empirical results on five datasets show that TGSRec significantly outperforms other baselines, in average up to 22.5% and 22.1% absolute improvements in Recall@10 and MRR, respectively. Our code is available online in https://github.com/DyGRec/TGSRec.

*Both authors contribute equally.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482242

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; **Recommender systems**; *Personalization.*

KEYWORDS

Sequential Recommendation, Temporal Effects, Graph Neural Network, Transformer

ACM Reference Format:
Ziwei Fan*, Zhiwei Liu*, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S. Yu. 2021. Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482242

1 INTRODUCTION

Recommender system has become essential in providing personalized information filtering services in a variety of applications [21, 26, 31, 40, 41]. It learns the user and item embeddings from historical records on the user-item interactions [8, 34]. In order to model the dynamics of the user-item interaction, current research works [5, 9, 35, 37, 42] leverage historical time-ordered item purchasing sequences to predict future items for users, referred to as the sequential recommendation (SR) problem [9, 12]. One of the fundamental assumptions of SR is that the users' interests change smoothly [9, 12, 37, 42]. Thus, we can train a model to infer the items more likely to appear in the future sequence. For example, with the recent developments of *Transformer* [38], current endeavors design a series of self-attention SR models to predict future item sequences [12, 36, 44]. A self-attention model infers sequence embeddings at position t by assigning an attention weight to each historical item and aggregating these items. The attention weights reveal impacts of previous items to the current state at time point t.

Despite their effectiveness, existing works only leverage the sequential patterns to model the item transitions within sequences, which is still insufficient to yield satisfactory results. The reason is that they ignore the crucial **temporal collaborative signals**, which are latent in evolving user-item interactions and coexist with

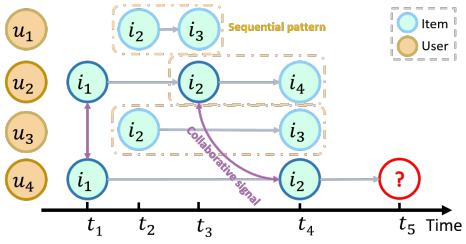
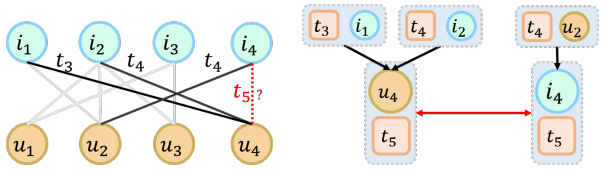


Figure 1: A toy example of temporal collaborative signals. Given the items that users u_1, u_2, u_3 and u_4 like in the past timestamps t_1, t_2, t_3 and t_4, the target is to recommend an item to u_4 at t_5 as the next item after i_2.

sequential patterns. To be specific, we present the effects of temporal collaborative signals in Figure 1. The target is to recommend an item to u_4 at t_5 as the next item after i_2. By only considering the sequential patterns, i_3 is recommended since it appears 2 times after i_2 as in u_1 and u_3, compared with i_4 of only 1 time in u_2. However, if also taking account of collaborative signals, we would recommend i_4, because both u_2 and u_4 have interactions with i_1 at t_2 and i_2 at t_3 and t_4, respectively, which indicates their high similarity. Hence, u_2's sequential patterns are of more impacts to u_4. This motivates us to *unify sequential patterns and temporal collaborative signals.*

However, incorporating temporal collaborative signals in SR is rather challenging. The first challenge is that it is hard to simultaneously encode collaborative signals and sequential patterns. Current models capture the sequential pattern based on the transition of items within sequences [9, 12, 28], thus lacking the mechanism to model the collaborative signals across sequences. Jodie [17] and DGCF [20] employs LSTM to model the dynamics and interactions of user and item embeddings but they cannot learn the impacts of all historical items, thus unable to encode sequences. SASRec [12] proposed to use a self-attention mechanism to encode item sequence, while the effects of users, i.e., collaborative signals, are omitted. SSE-PT [44] implicitly models collaborative signals by directly adding the same user embedding into the sequence encoding. However, it fails to model the interactions between user and item, thus unable to explicitly express collaborative signals.

The second challenge is that it is hard to express the temporal effects of collaborative signals. In other words, it remains unclear how to measure the impacts of those signals from a temporal perspective. For example, in Figure 1, i_1 is interacted with u_2 and u_4 at t_1, while i_2 is interacted with them respectively at t_3 and t_4. Since there is a lag, it is problematic to ignore the time gap and assume they are of equal contributions. We should use temporal information to infer the importance of those collaborative signals to the recommendation on t_5. Existing works [9, 12, 20, 36] assume that items appear discretely with equal time intervals. Thus, they only focus on the orders/positions of items in the sequence, which limits their capacity in expressing the temporal information. Some recent works [19, 50] also notice the importance of time span. But their models either fail to capture time differences between historical interactions or are unable to generalize to any unseen future timestamps or time difference, thus are still far from revealing the actual temporal effects of collaborative signals.



(a) Example of CTBG **(b) Temporal Inference**

Figure 2: The associated CTBG of Figure 1 and the inference of temporal embeddings of u_4 and i_4 at t_5.

Current transformer-based models [12, 36] adopt self-attention mechanism, which has query, key, and value inputs from item embeddings and employs dot-product to learn their correlation scores. The limitation is that self-attention is only able to capture item-item relationships in sequences. Additionally, they have no module to capture temporal correlations of items. To this end, we propose a new model **T**emporal **G**raph **S**equential **Rec**ommender (TGSRec). It consists of two novel components: (1) the Temporal Collaborative Transformer (TCT) layer and (2) graph information propagation.

The first component advances current transformer-based models as it can explicitly model collaborative signals in sequences and express temporal correlations of items in sequences. To be more specific, TCT layer adopts *collaborative attention* among user-item interactions, where the query input to the collaborative attention is from the target node (user/item), while the key and value inputs are from connected neighbors. As such, TCT layer learns the importance of those interactions, thus well characterizing the collaborative signals. Moreover, TCT layer fuses the temporal information into the collaborative attention mechanism, which explicitly expresses the temporal effects of those interactions. Altogether, the TCT layer captures temporal collaborative signals.

The second module is devised upon our proposed Continuous-Time Bipartite Graph (CTBG). The CTBG consists of user/item nodes, and interaction edges with timestamps, as shown in Figure 2a. Given timestamps, neighbor items of users preserve sequential patterns. We propagate temporal collaborative information learned around each node to surrounding neighbors over CTBG. Therefore, it unifies sequential patterns with temporal collaborative signals.

In this work, we propose to use temporal embeddings of nodes for recommendation, which are dynamic and inferred at specified timestamps. For example, at time t, we infer the temporal user embedding by aggregating the context. We illustrate the temporal inference of u_4 and i_4 at time t_5 in Figure 2b. The temporal embeddings are inferred by our proposed TCT layer. It uses temporal information to discriminate impacts of those historical interactions and makes inferences of temporal node embeddings. The contributions of this paper are as follows:

Graph Sequential Recommendation: We connect the SR problem with graph embedding methods, which focuses on unifying the sequential patterns and temporal collaborative signals.

Temporal Collaborative Transformer: We propose a novel temporal collaborative attention mechanism to infer temporal node embeddings, which jointly models collaborative signals and temporal effects. This overcomes the inadequacy of the traditional self-attention mechanism on capturing the temporal effects and user-item collaborative signals.

Extensive Experiments: We conduct a comparison experiment on five real-world datasets. Comprehensive experiments demonstrate the state-of-the-art performance of TGSRec and its effectiveness of modeling temporal collaborative signals.

2 RELATED WORK

In this section, we first review some related work, which includes sequential recommendation (SR), temporal information, and some graph-based recommender systems.

2.1 Sequential Recommendation

SR predicts the future items in the user shopping sequence by mining the sequential patterns. An initial solution to the SR problem is to build a Recurrent Neural Network (RNN) [9, 42, 51, 55]. GRU4Rec [9] is proposed to predict the next item in a session by employing the GRU modules. Later, a Hierarchical RNN [33] is proposed to enhance the RNN model regarding the personalizing information. Additionally, LSTM [10, 42] can be applied to explore both the long-term and short-term sequential patterns. Moreover, in order to capture the intent of users at local sub-sequence, NARM [18] is proposed by combining the RNN model with attention weights. The major drawback of the RNN model is that it can only generate a single hidden vector, which limits its power to encode sequences [2].

Recently, owing to the success of self-attention model [3, 22, 38, 52] in NLP tasks, a series of attention-based SR models are proposed [11, 12, 23, 28, 32, 36, 43]. SASRec [12] applies the transformer layer to assign weights to items in the sequence. Later, inspired by the BERT [3] model, BERT4Rec [36] is proposed with a bidirectional transformer layer. [28] introduce the sequence to sequence training procedure in SR. SSE-PT [44] designs a personalized transformer to improve the SR performance. ASReP [23] proposes augmenting short sequences to alleviate the cold-start issue in Transformer. TiSASRec [19] enhances SASRec with the time-interval information found in the training data. However, these models only focus on the item transitions within sequences, while unable to unify the important temporal collaborative signals with sequential patterns and are not generalized to unseen timestamps.

2.2 Temporal Information

Previously mentioned SR works are specifically designed to capture sequential patterns, while ignoring the important temporal information [15, 17, 19, 46, 47]. In practice, the context of users and items changes over time, which is crucial for modeling the temporal dynamics in SR. TimeSVD++ [15] is a representative work which models the temporal information into collaborative filtering (CF) method. It simply treats the bias as a function over time. BPTF [47] extends matrix factorization to tensor factorization and uses time as the third dimension. MS-IPF [46] defines a temporal graph, where it operates PageRank algorithm for recommendation. Recently, CDTNE [30] is proposed by applying temporal random walk over its defined continuous-time dynamic network. TGAT [49] also introduces temporal attention for learning dynamic graph embeddings. JODIE [17] develops user and item RNNs to update user and item embeddings. Regarding the SR problem, a few recent works [19, 50] also notice the importance of temporal information. CTA [43], MTAM [11], and TiSASRec [19] all consider to use time intervals between successive items in sequences. TASER [50] encodes both the absolute time and relative time as vectors, which are processed to attention models to complete the SR task. However, these models are not able to unify temporal collaborative signals with sequential patterns.

2.3 Graph-based Recommendation

Because we solve the SR problem based on the graph structure [53, 54], we also review some graph-based recommender system models [7, 24–26, 30, 40], especially those based on Graph Neural Network (GNN) methods [1, 7, 14, 40]. Compared with directly learning from sequences, graph-based models can also capture the structural information [1, 30]. Both NGCF [40] and LightGCN [7] argue that graph-based models are able to effectively model collaborative signals, which is crucial in learning user/item embeddings. The successes of GNN in recommender systems [1, 7, 39, 40] provide simple yet effective methods in learning user/item embeddings from graphs. GNN models learn the embeddings by aggregating neighbors [7, 40]. Therefore, it is easy to stack multiple layers to learn both the first-order and high-order collaborative signals [7, 40]. CTDNE [30] defines a temporal graph to learn dynamic embeddings of nodes. TGAT [49] learns the dynamic graph embeddings based on the graph attention model. Basconv [25] characterizes heterogeneous graphs to learn user/item embeddings. Those models argue that graph is powerful in modeling both the structural and temporal information. However, few works investigate the possibility of solving SR problems based on graphs. SR-GNN [45] learns embeddings of session graphs by using a GNN to aggregate item embeddings but fails to model temporal collaborative signals.

3 DEFINITIONS AND PRELIMINARIES

In this section, we introduce some definitions and preliminaries. Different from using users' interactions sequences as inputs in SR, we introduce the *Continuous-Time Bipartite Graph* (CTBG) to represent all temporal interactions. Each edge in this graph has the timestamp as the attribute. The directly connected neighbors of every user/item node in this graph preserve the sequential order via the timestamps at edges. The formal definition of CTBG are given in the following:

Definition 3.1 (Continuous-Time Bipartite Graph). *A continuous time bipartite graph with N nodes and E edges for recommendations is defined as $\mathcal{B} = \{\mathcal{U}, \mathcal{I}, \mathcal{E}_\mathcal{T}\}$, where $\mathcal{U}$ and $\mathcal{I}$ are two disjoint node sets of users and items, respectively. Every edge $e \in \mathcal{E}_\mathcal{T}$ is denoted as a tuple $e = (u, i, t)$, where $u \in \mathcal{U}$, $i \in \mathcal{I}$, and $t \in \mathbb{R}^+$ as the edge attribute. Each triplet (u, i, t) denotes the interaction of a user u with item i at timestamp t.*

This paper focuses on the SR problem with continuous timestamps. An example of the CTBG is presented in Figure 2a. Let $\mathcal{I}_u(t)$ denote the set of items interacted with the user u before timestamp t, and $\mathcal{I} \setminus \mathcal{I}_u(t)$ denote the remaining items. We defined the continuous-time sequential recommendation problem which we study in this paper as following:

Definition 3.2 (Continuous-Time Recommendation). *At a specific timestamp t, given user set $\mathcal{U}$, item set $\mathcal{I}$, and the associated CTBG, the continuous-time recommendation of u is to generate a*

ranking list of items from $\mathcal{I} \setminus \mathcal{I}_u(t)$, where the items that u is interested will be ranked top in the list.

Then, the SR problem is equivalent to make continuous-time recommendations on a set of future timestamps $\mathcal{T}_u$ for each user u:

Definition 3.3 (Continuous-Time Sequential Recommendation). *For a specific user u, given a set of future timestamps $\mathcal{T}_u > T$, the continuous-time sequential recommendation for this user is to make a continuous-time recommendation for every timestamp $t \in \mathcal{T}_u$.*

This is a generalized definition compared with other works [12, 28]. We explicitly consider timestamps, while others only care about the orders/positions. Therefore, differing from existing works using next-item prediction to evaluate sequential recommendation, future timestamps should be present to make a prediction. If timestamps are position numbers in sequences, the studied problem is reduced to the same definition as using only orders/positions information. Note that timestamp can be any real value, thus being continuous.

4 PROPOSED MODEL

In this section, we present the TGSRec model, which unifies sequential patterns and temporal collaborative signals. The framework of the TGSRec model is presented in Figure 3. There are three major components: 1) Embedding layer, which encodes nodes and timestamps in a consistent way to connect the SR problem with graph embedding method; 2) Temporal Collaborative Transformer (TCT) layer, which employs a novel temporal collaborative attention mechanism to discriminate temporal impacts of neighbors, and aggregates both node and time embeddings to infer the temporal node embedding; 3) Prediction layer, which utilizes output embeddings from the final TCT layer to calculate the score.

4.1 Embedding Layer

We encode two types of embeddings in this paper, one being the *long-term embeddings* of nodes, and the other being the *continuous-time embeddings* of timestamps on edges.

4.1.1 Long-Term User/Item Embeddings. Long-term embeddings for users and items are necessary [4] for long-term collaborative signals representation. In CTBG, it functions as node features and is optimized to model the holistic structural information. A user (item) node is parameterized by a vector $\boldsymbol{e}_u(\boldsymbol{e}_i) \in \mathbb{R}^d$. Since we learn embeddings for nodes in the CTBG, we retrieve the embedding of a node by indexing an embedding table $\boldsymbol{E} = [\boldsymbol{E}_{\mathcal{U}}; \boldsymbol{E}_{\mathcal{I}}] \in \mathbb{R}^{d \times |\mathcal{V}|}$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$. Note that the embedding table $\boldsymbol{E}$ serves as a starting state for the inference of temporal user/item embeddings. During the training process, $\boldsymbol{E}$ will be optimized.

4.1.2 Continuous-Time Embedding. The continuous time encoding [48, 50] behaves as a function that maps those scalar timestamps into vectors, i.e., $\Phi : T \mapsto \mathbb{R}^{d_T}$, where $T \in \mathbb{R}^+$. Based on previous SR models [19, 43, 50], time span plays a vital component in expressing the temporal effects and uncovering sequential patterns. The time encoding function embeds timestamps into vectors so as to represent the time span as the dot product of corresponding encoded time embeddings. Therefore, we define the temporal effects as a function of time span in continuous time space: given a pair of interactions (u, i, t_1) and (u, j, t_2) of the same user, the **temporal effect** is defined as a function $\psi(t_1 - t_2) \mapsto \mathbb{R}$, which is expressed as a kernel value of the time embeddings of t_1 and t_2:

$$\psi(t_1 - t_2) = \mathcal{K}(t_1, t_2) = \Phi(t_1) \cdot \Phi(t_2), \tag{1}$$

where $\mathcal{K}$ is the temporal kernel and $\cdot$ denotes the dot product operation. The temporal effect $\psi(t_1 - t_2)$ measures the temporal correlation between two timestamps. Moreover, the time encoding function should be generalized to any unseen timestamp such that any time span not found in training data can still be inferred by the encoded time embeddings. Unlike modeling the absolute time difference like [19], representing temporal effects as a kernel is generalized to any timestamp as it models the time representations directly. Therefore, the temporal effect of any pair of timestamps can be inductively inferred by the dot product of time representations. Eq. (1) can be achieved by a continuous and translation-invariant kernel $\mathcal{K}(t_1, t_2)$ based on Bochner's Theorem [27]. By explicitly representing the temporal features, the temporal embedding is:

$$\Phi(t) \mapsto \sqrt{\frac{1}{d_T}} \left[\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_{d_T} t), \sin(\omega_{d_T} t)\right]^\top, \tag{2}$$

where $\boldsymbol{\omega} = \left[\omega_1, \ldots, \omega_{d_T}\right]^\top$ are learnable and d_T is the dimension.

4.2 Temporal Collaborative Transformer

Next, we present the novel TCT layer of TGSRec. We intend to address two strengths of a TCT layer: (1) constructing information from both user/item embeddings and temporal embedding, which explicitly characterizes temporal effects of the correlations; (2) a collaborative attention module, which advances existing self-attention mechanism by modeling the importance of user-item interactions, which is thus able to explicitly recognize collaborative signals. To achieve this, we first present the information construction and aggregation from a user perspective. Then, we introduce a novel collaborative attention mechanism to infer importance of interactions. Finally, we demonstrate how to generalize to items.

4.2.1 Information Construction. We construct input information of each TCT layer as the combination of long term node embeddings and time embeddings. As such, we can unify temporal information and collaborative signals. In particular, the *query* input information at the l-th layer for user u at time t is:

$$\boldsymbol{h}_u^{(l-1)}(t) = \boldsymbol{e}_u^{(l-1)}(t) \| \Phi(t), \tag{3}$$

where $l = 1, 2, \ldots, L$. $\boldsymbol{h}_u^{(l-1)}(t) \in \mathbb{R}^{d+d_T}$ is the information for u at t, $\boldsymbol{e}_u^{(l-1)}(t) \in \mathbb{R}^d$ is the temporal embedding of u, and $\Phi(t) \in \mathbb{R}^{d_T}$ denotes the time vector of t. $\|$ denotes the concatenation operation. Other operations including summation are possible. However, we use concatenation for simplicity. It also provides intuitive interpretation in the attention, as shown in Eq. (7). Note that when $l = 1$, it is the first TCT layer. The temporal embedding $\boldsymbol{e}_u^{(0)}(t) = \boldsymbol{E}_u$, i.e., the long-term user embedding. When $l > 1$, the temporal embedding is generated from the previous TCT layer.

In addition to the query node itself, to we also propagate temporal collaborative information from its neighbors. We randomly sample S different interactions of u before time t as $\mathcal{N}_u(t) = \{(i, t_s) | (u, i, t_s) \in \mathcal{E}_t \text{ and } t_s < t\}$. The input information at the l-th layer for each

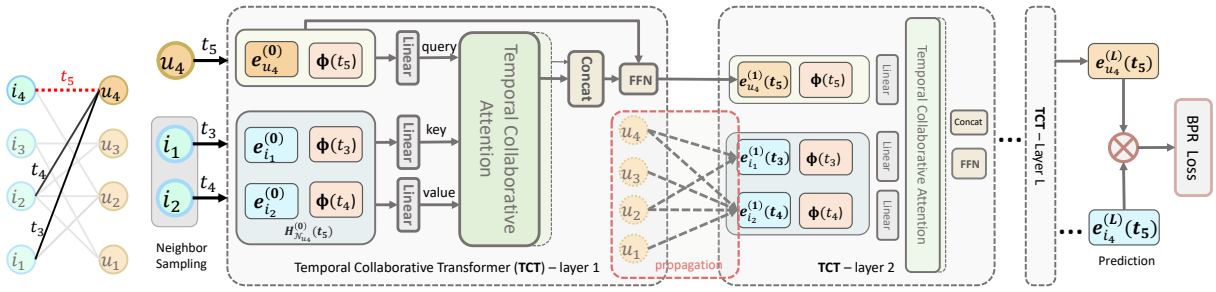


Figure 3: The framework of TGSRec. The query node is u_4, whose final temporal embedding at time t_5 is $\boldsymbol{h}_{u_4}^{(2)}(t_5)$. The TCT layer samples its neighbor nodes and edges. Timestamps on edges are encoded as vectors by using mapping function Φ. Node embeddings for the first TCT layer are long-term embeddings. Node embeddings for other TCT layers (*e.g.* layer 2) are propagated from the previous TCT layer, thus being temporal node embeddings.

(i, t_s) pair is:

$$\boldsymbol{h}_i^{(l-1)}(t_s) = \boldsymbol{e}_i^{(l-1)}(t_s) \| \Phi(t_s), \tag{4}$$

where $\boldsymbol{h}_i(t_s)$ is the information for item i at t_s, $\boldsymbol{e}_i(t_s)$ denotes the temporal embedding of i at t_s. Again, note that when $l = 1$, $\boldsymbol{e}_i^{(0)}(t_s) = \boldsymbol{E}_i$, i.e., the long-term item embedding. When $l > 1$, the temporal embedding is output from the previous TCT layer.

4.2.2 Information Propagation. After constructing the information, we propagate the information of sampled neighbors $\mathcal{N}_u(t)$ to infer the temporal embeddings. Since the neighbors are involving with time t, in this way, we can unify the sequential patterns with temporal collaborative signals. We compute the linear combination of the information from all sampled interactions as:

$$\boldsymbol{e}_{\mathcal{N}_u}^{(l)}(t) = \sum_{(i,t_s)\in N_u(t)} \pi_t^u(i, t_s) \boldsymbol{W}_v^{(l)} \boldsymbol{h}_i^{(l-1)}(t_s), \tag{5}$$

where $\pi_t^u(i, t_s)$[1] denotes the importance of an interaction (u, i, t_s) and $\boldsymbol{W}_v \in \mathbb{R}^{d\times(d+d_T)}$ is the linear transformation matrix. $\pi_t^u(i, t_s)$ represents the impact of a historical interaction (u, i, t_s) to the temporal inference of u at time t, which is calculated by the temporal collaborative attention.

4.2.3 Temporal Collaborative Attention. We adopt the novel temporal collaborative attention mechanism to measure the weights $\pi_t^u(i, t_s)$, which considers both neighboring interactions and the temporal information on edges. Both factors contribute to the importance of historical interactions. Thus, it is a better mechanism to capture temporal collaborative signals than self-attention mechanism that only models item-item correlations. The attention weight $\pi_t^u(i, t_s)$ is formulated as follows:

$$\pi_t^u(i, t_s) = \frac{1}{\sqrt{d + d_T}} \left(\boldsymbol{W}_k^{(l)} \boldsymbol{h}_i^{(l-1)}(t_s)\right)^\top \boldsymbol{W}_q^{(l)} \boldsymbol{h}_u^{(l-1)}(t), \tag{6}$$

where $\boldsymbol{W}_k^{(l)}$ and $\boldsymbol{W}_q^{(l)}$ are both linear transformation matrices, and the factor $\frac{1}{\sqrt{d+d_T}}$ protects the dot-product from growing large with high dimensions. We adopt dot-product attention because if we ignore transformation matrices and the scalar factor, based on Eq. (3) and Eq. (4), the right-hand side of Eq. (6) can be rewritten as:

$$\boldsymbol{e}_u^{(l-1)}(t) \cdot \boldsymbol{e}_i^{(l-1)}(t_s) + \Phi(t) \cdot \Phi(t_s), \tag{7}$$

where the first term denotes the user-item collaborative signal, and the second term models the temporal effect according to Eq. (1). With more stacked layers, collaborative signals and temporal effects are entangled and tightly connected. Hence, the dot-product attention can characterize impacts of temporal collaborative signals.

Hereafter, we normalize the attention weights across all sampled interactions by employing a softmax function:

$$\pi_t^u(i, t_s) = \frac{\exp\left(\pi_t^u(i, t_s)\right)}{\sum_{(i',t'_s)\in\mathcal{N}_u(t)} \exp\left(\pi_t^u(i', t'_s)\right)}. \tag{8}$$

Moreover, the computation is implemented by packing the information of all sampled interactions. To be more specific, we stack the information (Eq. (4)) of all sampled interactions as a matrix $\boldsymbol{H}_{\mathcal{N}_u}^{(l-1)}(t) \in \mathbb{R}^{(d+d_T)\times S}$, as illustrated[2] in Figure 3. We denote $\boldsymbol{K}_u^{(l-1)}(t) = \boldsymbol{W}_k^{(l)} \boldsymbol{H}_{\mathcal{N}_u}^{(l-1)}(t)$, $\boldsymbol{V}_u^{(l-1)}(t) = \boldsymbol{W}_v^{(l)} \boldsymbol{H}_{\mathcal{N}_u}^{(l-1)}(t)$ and $\boldsymbol{q}_u^{(l-1)}(t) = \boldsymbol{W}_q^{(l)} \boldsymbol{h}_u^{(l-1)}(t)$, which are respectively the key, value and query input for the temporal collaborative attention module. We illustrate this in Figure 3 as green blocks. For simplicity and without ambiguity, we ignore the superscripts and time t and combine Eq. (6) and Eq. (8). Then, we can rewrite the Eq. (5) as:

$$\boldsymbol{e}_{\mathcal{N}_u} = \boldsymbol{V}_u \cdot \text{Softmax}\left(\frac{\boldsymbol{K}_u^\top \boldsymbol{q}_u}{\sqrt{d + d_T}}\right), \tag{9}$$

which is in the form of dot-product attention in Transformer [38]. Therefore, we can safely apply the multi-head attention operation and concatenate the output from each head as the information for aggregation, which is presented in Figure 3. Note that our attention is not a self-attention but a temporal collaborative attention, which jointly models user-item interactions and temporal information.

4.2.4 Information Aggregation. To output the temporal node embedding, the final step of a TCT layer is to aggregate the query information in Eq. (3) and the neighbor information in Eq. (5). We concatenate and send them to a feed-forward neural network (FFN):

$$\boldsymbol{e}_u^{(l)}(t) = \text{FFN}\left(\boldsymbol{e}_{\mathcal{N}_u}^{(l)}(t) \| \boldsymbol{h}_u^{(l-1)}(t)\right), \tag{10}$$

[1] $\pi_t^u(i, t_s)$ also has a superscript of the layer number l, which is ignored for simplicity.

[2] In figure 3, an embedding is a row vector, while in notations, it is a column vector.

where $\boldsymbol{e}_u^{(l)}(t)$ is the temporal embedding of u at t on l-th layer, and FFN consists of two linear transformation layers with a ReLU activation function in between [38]. The output temporal embedding $\boldsymbol{e}_u^{(l)}(t)$ can either be sent to the next layer or output as the final temporal node embedding for prediction.

4.2.5 Generalization to items. Though we only present the TCT layer from the user query perspective, it is analogous if the query is an item at a specific time. We only need to alternate the user query information to the item query information, and change the neighbor information in Eq. (4) and Eq. (5) accordingly as user-time pairs. Then, we can make an inference of the temporal embedding of item i at time t as $\boldsymbol{e}_i^{(l)}(t)$, which is sent to the next layer.

4.3 Model Prediction

The TGSRec model consists of L TCT layers. For each test triplet (u, i, t), it yields temporal embeddings for both u and i at t on the last TCT layer, denoting as $\boldsymbol{e}_u^{(L)}(t)$ and $\boldsymbol{e}_i^{(L)}(t)$, respectively. Then, the prediction score is:

$$r(u, i, t) = \boldsymbol{e}_u^{(L)}(t) \cdot \boldsymbol{e}_i^{(L)}(t), \tag{11}$$

where $r(u, i, t)$ denotes the score to recommend i for u at time t. With the generalized continuous-time embeddings and the proposed TCT layers, we can generalize and infer user/item embeddings at any timestamp, thus making multiple steps recommendation feasible while existing work only predicts next item. Recall that based on the Definition 3.3, we recommend each user a ranking list of items at the given timestamp. Therefore, we use Eq. (11) to calculate scores of all candidate items and sort them by scores.

4.4 Model Optimization

To learn the model parameters, we use the pairwise BPR loss [34], which is widely used for top-N recommendation. The pairwise BPR loss assumes that the observed implicit feedback items have greater prediction scores than those unobserved and is also designed for ranking based top-N recommendation. The objective function is:

$$\mathcal{L}_{bpr} = \sum_{(u,i,j,t) \in O_T} -\log\sigma\left(r(u, i, t) - r(u, j, t)\right) + \lambda||\Theta||_2^2, \tag{12}$$

where O_T denotes the training samples, Θ includes all learnable parameter, and $\sigma(\cdot)$ is a sigmoid function. The training samples $O_T = \{(u, i, j, t)|(u, i, t) \in \mathcal{E}_T, j \in \mathcal{I} \setminus \mathcal{I}_u(t)\}$, where the positive interaction (u, i, t) comes from the edge set $\mathcal{E}_T$ of CTBG, the negative item j is sampled from unobserved items $\mathcal{I} \setminus \mathcal{I}_u(t)$ of user u at timestamp t; Θ includes long-term embedding E, time embedding parameter ω, and all linear transformation matrices. The loss is optimized via mini-batch Adam [13] with adaptive learning rate. Alternatively, we can optimize the model with a Binary Cross Entropy (BCE) loss as:

$$\mathcal{L}_{bce} = \sum_{(u,i,j,t) \in O_T} \log\sigma\left(r(u, i, t)\right) + \log\sigma\left(1 - r(u, j, t)\right) + \lambda||\Theta||_2^2, \tag{13}$$

which is compared with BPR loss in experiments.

5 EXPERIMENTS

In this section, we present the experimental setups and results to demonstrate the effectiveness of TGSRec. The experiments answer the following Research Questions (RQs):

- **RQ1**: Does TGSRec yield better recommendation?
- **RQ2**: How do different hyper-parameters (*e.g.,* number of neighbors S, etc.) affect the performance of TGSRec?
- **RQ3**: How do different modules (*e.g.,* temporal collaborative attention, etc.) affect the performance of TGSRec?
- **RQ4**: Can TGSRec effectively unify sequential patterns and temporal collaborative signals? (Reveal temporal correlations)

5.1 Datasets

We conduct our experiments on four Amazon review datasets [29] and MovieLens ML-100K dataset [6]. The Amazon datasets are collected from different domains[3], from the Amazon website during May 1996 to July 2014. The Movie Lens dataset is collected from September 19th, 1997 through April 22nd, 1998. We use Unix timestamps on all datasets. For each dataset, we chronologically split for train/validation/test in 80%/10%/10% ratio based on the interaction timestamps. More details, such as data descriptions and statistics, are presented in the Table 1. We can find amazon datasets are much sparser and their time spans are much longer compared with ML-100K dataset. For Amazon related datasets, the time intervals of successive interactions are typically in days, while ML-100k has shorter time intervals, ranging from seconds to days.

Table 1: Statistics of datasets.

Dataset	Toys	Baby	Tools	Music	ML100K
#Users	17,946	17,739	15,920	4,652	943
#Items	11,639	6,876	10,043	3,051	1,682
#Edges	154,793	146,775	127,784	54,932	48,569
#Train	134,632	128,833	107,684	51,765	80,003
#Valid	11,283	10,191	10,847	2,183	1,516
#Test	8,878	7,751	9,253	984	1,344
Density	0.07%	0.12%	0.08%	0.38%	6.30%
Avg. Int.	85 days	61 days	123 days	104 days	4.8 hours

"Av. Int." denotes average time interval.

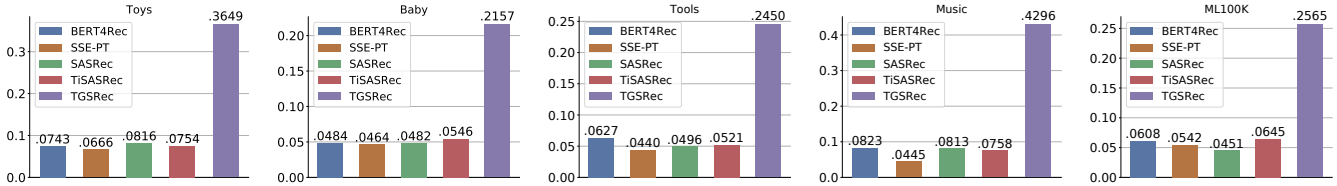
5.2 Experimental Settings

5.2.1 Baselines. We compared TGSRec with the state-of-the-art methods in three different groups. **Static models**: Static models ignore the temporal information and generate static user/item embeddings for a recommendation. We compare with the most standard baseline BPRMF [34], and also compare with a recent GNN-based model LightGCN [7]. **Temporal models**: We compare some relevant temporal methods, such as CTDNE [30] and one recent model TiSASRec [19], which utilize time information. We also try to compare with JODIE [17]. However, we do not report it because has out-of-memory errors on most datasets. **Transformer-based SR models**: Since our model is built upon the transformer, we mainly focus on comparing with the recent transformer-based SR methods,

[3] https://jmcauley.ucsd.edu/data/amazon/

Table 2: Overall Performance w.r.t. Recall@{10,20} and MRR.

Datasets	Metric	BPR	LightGCN	SR-GNN	GRU4Rec	Caser	SSE-PT	BERT4Rec	SASRec	TiSASRec	CDTNE	TGSRec	Improv.
Toys	Recall@10	0.0021	0.0016	0.0020	0.0274	0.0138	0.1213	0.1273	0.1452	0.1361	0.0016	**0.3650**	0.2198
	Recall@20	0.0036	0.0026	0.0033	0.0288	0.0238	0.1719	0.1865	0.2044	0.1931	0.0045	**0.3714**	0.1670
	MRR	0.0024	0.0018	0.0018	0.0277	0.0082	0.0595	0.0643	0.0732	0.0671	0.0025	**0.3661**	0.2929
Baby	Recall@10	0.0028	0.0036	0.0030	0.0036	0.0077	0.0911	0.0884	0.0975	0.1040	0.0218	**0.2235**	0.1195
	Recall@20	0.0039	0.0045	0.0062	0.0048	0.0193	0.1418	0.1634	0.1610	0.1662	0.0292	**0.2295**	0.0663
	MRR	0.0019	0.0024	0.0024	0.0028	0.0071	0.0434	0.0511	0.0455	0.0521	0.0157	**0.2147**	0.1626
Tools	Recall@10	0.0023	0.0021	0.0051	0.0048	0.0077	0.0775	0.1296	0.0913	0.0946	0.0186	**0.2457**	0.1161
	Recall@20	0.0036	0.0035	0.0092	0.0059	0.0161	0.1155	0.1784	0.1337	0.1356	0.0271	**0.2559**	0.0775
	MRR	0.0026	0.0023	0.0028	0.0051	0.0068	0.0419	0.0628	0.0460	0.0480	0.0203	**0.2468**	0.1840
Music	Recall@10	0.0122	0.0142	0.0051	0.0549	0.0183	0.0915	0.1352	0.1372	0.1372	0.0071	**0.5935**	0.4563
	Recall@20	0.0152	0.0183	0.0092	0.0589	0.0346	0.1494	0.2093	0.2094	0.1951	0.0163	**0.5986**	0.3892
	MRR	0.0057	0.0064	0.0028	0.0540	0.0106	0.0423	0.0824	0.0768	0.0681	0.0037	**0.3820**	0.2996
ML100k	Recall@10	0.0461	0.0565	0.0045	0.0996	0.0246	0.1079	0.1116	0.09450	0.1332	0.0350	**0.3118**	0.1786
	Recall@20	0.0766	0.0960	0.0060	0.1168	0.0417	0.1801	0.1786	0.1808	0.2232	0.0536	**0.3252**	0.1020
	MRR	0.0213	0.0252	0.0012	0.0938	0.0147	0.0519	0.0600	0.0448	0.0605	0.0162	**0.2416**	0.1478



(a) NDCG@10 in Toys **(b) NDCG@10 in Baby** **(c) NDCG@10 in Tools** **(d) NDCG@10 in Music** **(e) NDCG@10 in ML100K**

Figure 4: NDCG@10 Performance in all Datasets. We ignore other methods because of their low values.

which are SASRec [12], BERT4Rec [36], SSE-PT [44], and TiSASRec [19]. **Other SR models**: In addition, we also compare with other type of SR models, i.e., FPMC [35], GRU4Rec [9], Caser [37], and SR-GNN [45], for comprehensive study.

For each testing interaction (u, i, t_{test}), our continuous-time sequential recommendation setting allows models to use any history interactions $\{(u, i, t)|t < t_{test}\}$ during the prediction stage, regardless of whether the historical interactions are in training portion, validation part or even in testing set. However, all parameters of models are only learned from the training data.

We implement TGSRec with Pytorch in a Nvidia 1080Ti GPU. We grid search all parameters and report the test performance based on the best validation results. For all models, we search for the dimensions of embeddings d in range of $[8, 16, 32, 64]$ and we tune the learning rate in $[10^{-2}, 10^{-3}, 10^{-4}]$, search the L2 regularization weight from $[5 \times 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}]$. For sequential methods, we search the maximum length of sequence in $[50, 100]$, number of layers from $[1, 2, 3]$, and number of heads in $[1, 2, 4]$.

5.2.2 Evaluation Protocol. All models will generate a ranking list of items for each testing interaction. Each evaluation metric is averaged over the total number of interactions as the final reported result. In order to accelerate the evaluation, we sample 1,000 negative items for evaluation instead of full set of negative items. For each interaction (u, i, t) in validation/test sets, we treat items that u has no interactions with before t as negative items. Regarding the sampling bias for evaluation [16], we apply the unbiased estimator in [16] to correct the sampled ranks. We evaluate the *top-N* recommendation performance by standard ranking-based evaluation metrics *Recall@N*, *NDCG@N*, and *Mean Reciprocal Rank (MRR)*. We set N to be either 10 or 20 for a comprehensive comparison.

5.3 Performance Comparison (RQ1)

We compare the performance of all models and demonstrate the superiority of TGSRec. We report the Recall and MRR of all models in Table 2. Additionally, we visualize the comparisons of NDCG in Figure 4. We have the following observations:

- TGSRec consistently and significantly outperforms all baselines in all datasets. In particular, for absolute performance improvement gains relative to the 2nd best, TGSRec achieves 22.51%, 16.90% and 22.15% absolute gains at recall@10, recall@20, and MRR, respectively. TGSRec also significantly outperforms others in NDCG, as shown in Figure 4. Several factors together determine the superiority of TGSRec: (1) TGSRec captures temporal collaborative signals; (2) TGSRec explicitly expresses temporal effects; and (3) TGSRec stacks multiple TCT layers to propagate the information.
- Those static methods achieve the worst performance among all models. A simple GRU4Rec even performs 10 times better than them. This indicates that static embeddings fail to utilize the temporal information, limiting its recommendation ability in SR. Thus, it is important to model dynamics.
- The CDTNE performs better than Caser and GRU4Rec in Tools and Baby datasets. This suggests the benefit of modeling temporal

Table 3: Ablation analysis (Recall@10) on five datasets. Bold score indicates performance better than the default version, while ↓ indicates a performance drop more than 50%.

Architecture	Toys	Baby	Tools	Music	ML100K
(0) Default	**0.3649**	**0.2235**	**0.3623**	**0.5935**	0.3118
(1) Mean	0.0027↓	0.0210↓	0.0055↓	0.0051↓	0.0647↓
(2) LSTM	0.0991↓	0.1237	0.1266↓	0.3740	0.3088
(3) Fixed ω	0.0854↓	0.0944↓	0.0910↓	0.3679	0.2789
(4) Position	0.0380↓	0.0243↓	0.0209↓	0.0742↓	0.0878↓
(5) Empty	0.0139↓	0.0240↓	0.0018↓	0.0346↓	0.0603↓
(6) BCELoss	0.2200	0.1916	0.1763↓	0.4624	**0.3542**

information with a graph. But it is still much worse than those transformer-based methods, which again proves the strength of transformer in encoding sequences. We also notice the poor performance of SR-GNN. We analyze the data and find time intervals between successive interactions vary a lot. Since SR-GNN is originally designed for session-based sequences, it is not suitable for SR with a long time span.

- The transformer-based SR methods consistently outperform all other types of baselines, which demonstrates the effectiveness of using transformer structure to encode sequence. Among them, TiSASRec is better than SASRec on two datasets, which proves the effectiveness of using time information. But it is still far worse than TGSRec. The reason is twofold. One is that only the interval information is not enough to unify the temporal information with sequential patterns. The other is that the proposed temporal collaborative attention in TCT layer captures more precise and generalized temporal effects. We find that BERT4Rec is better than the other baselines on the Tools dataset but not better on other datasets. Since the main difference between BERT4Rec and SASRec is the bi-directional sequence encoding, it may break causal relations among items within a sequence. The TGSRec performs much better than SR models, showing the necessity of unifying sequential patterns and temporal collaborative signals.

5.4 Parameter Sensitivity (RQ2)

In this section, we conduct sensitivity analyses of the hyperparameters of TGSRec, including the number of layers L, embedding size d, and the number of neighbors S. The results are reported in Figure 5.

The number of layers. The number of TCT layers L is searched from $\{0, 1, 2\}$. The results are shown in the top row of Figure 5. When $L = 0$, TGSRec has no TCT layer, thus unable to infer temporal embeddings. We can observe it performs the worst on all dataset, which justify the benefit of temporal inference. When $L = 1$, it makes temporal inference, but without propagation to the next layer. Therefore, it still performs worse than $L = 2$ on most datasets. When $L = 2$, it can not only make temporal inference, but also propagate the information to capture high-order signals, which alleviates the data sparsity problem.

Embedding size. The embedding size d of TCT layers is searched from $\{8, 16, 32, 64\}$, which is presented at the mid-row in Figure 5. We can find that the performance increases as the embedding size enlarges. However, when the embedding size is too large, *e.g.*, $d = 64$, the performance drops, which results from the over-fitting problem because of too many parameters.

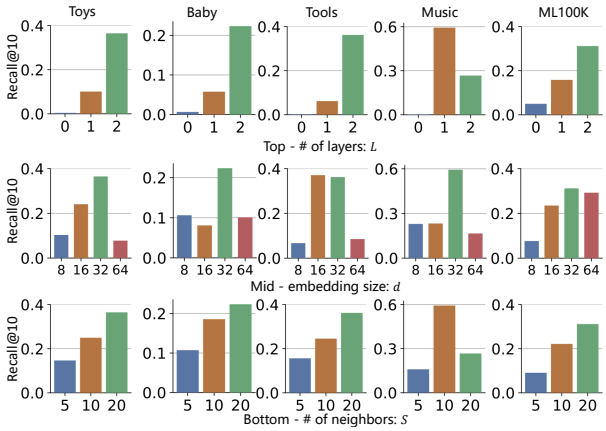


Figure 5: Recall@10 w.r.t. L, d and S on 5 datasets.

Number of neighbors. The number of neighbors S is searched in $\{5, 10, 20\}$, which is illustrated in the bottom row of Figure 5. We can observe that TGSRec has performance gains on most datasets as the number of neighbors grows. It is because more neighbors can provide more information for encoding both sequences and temporal collaborative signals.

5.5 Ablation Study (RQ3 & RQ4)

In this section, we conduct experiments to analyze different components in TGSRec. We develop several variants to better understand their effectiveness. Table 3 shows the performance w.r.t. Recall@10 of the default TGSRec and other variants. We label each row with an index number for quick reference. The default is TGSRec with all components and labeled as 0. We develop the variants by substituting some components, which are temporal collaborative attention (1-2), continuous-time embeddings (3-5), and loss function (6):

Temporal collaborative attention. We replace the proposed temporal collaborative attention of sampled neighbors with a mean pooling or LSTM module, both of which are widely used to encode sequences. Results are labeled as (1) and (2) in Table 3. We can observe that substituting collaborative attention with a mean pooling layer severely spoils the performance. Compared with that, the adoption of LSTM is much better, indicating the necessity of encoding sequential patterns by considering item transitions. However, both of them are worse than the default one, which implies the advantage of temporal collaborative attention in encoding sequences.

Continuous-time embedding. We use three variants to verify the efficacy of the time mapping function Φ. The first variant is that we sample ω in Eq. (2) directly from a normal distribution. The second and third variants replace the Φ with a learnable positional embedding as in [12] and emptying all zeros, respectively. The results are labeled as (3) – (5) in Table 3. Because of the better performance of position embedding compared with empty embedding, we can conclude that TGSRec has the ability to encode sequential patterns. In addition, we also find that even a fixed ω to learn

Table 4: Variants of Temporal Information Construction

Variant	Toys	Baby	Tools	Music	ML100K
TGSRec	**0.3649**	**0.2235**	**0.3623**	**0.5935**	**0.3118**
$\mathcal{U}$ w/o T	0.0103	0.0138	0.0106	0.0112	0.1555
$\mathcal{I}$ w/o T	0.1013	0.0961	0.0836	0.2785	0.2336

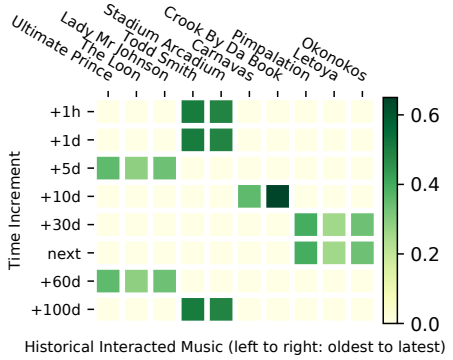


Figure 6: Temporal Attention Weights Visualization

the time embedding can significantly outperform the position embedding, indicating the necessity of using the temporal kernel to capture temporal effects in sequences. Moreover, the default version, using a trainable ω, achieves the best performance, which indicates its capacity to learn temporal effects from data.

Loss function. We also compare BPR loss and BCE Loss, which is labeled as (6) in Table 3. The results indicate that the BCE loss performs inferior to BPR loss, except for the ML100K dataset. This is because BPR loss is optimized for ranking while BCE loss is designed for binary classification.

5.6 Temporal Correlations (RQ4)

Though we have already indicated the answer of RQ4 in Sec. 5.5, this section also conducts detailed analyses of the temporal correlation within sequences to directly answer **RQ4**.

5.6.1 Temporal Information Construction. We develop two variants by dismissing the time vector in either Eq. (3) or Eq. (4), i.e., users without time vectors or items without time vectors. The results are presented in Table 4. The observations are two-fold. Firstly, the performance of items without time is better than users without time. It implies that the temporal inference of user embeddings are rather important, which matches the intuition that the preference of users are dynamic while items are relatively more static. Secondly, the performance deteriorates significantly in both variants, indicating again TGSRec is able to model temporal effects of collaborative signals while also encoding sequences.

5.6.2 Temporal Attention Weights Visualization. We visualize the attention weights of TGSRec on the Music dataset for a user, which is shown in Figure 6. Each row is associated with an increment ('h' for hour and 'd' for day) from the last interactive timestamp T = 1159142400, where 'next' denotes the timestamp (T+34d) for the test interaction. Each column is associated with an item. We can observe that the attention weights for items are dynamic at different timestamps, which indicates the temporal inference characteristics of TGSRec. Moreover, the time increments can be arbitrary values, which verifies its continuity.

Table 5: Recommendation w.r.t. time increments after the last interaction at timestamp T = 1159142400. 'next' is the timestamp of the test interaction. The ground truth item is in red color. Items also predicted by SASRec and TiSASRec are in blue color.

Time	Rank-1	Rank-2	Rank-3	Rank-4
T+5d	Letoya	H. of Blue L.	Ult. Prince	Veneer
T+30d	J. of A Gemini	Living Lgds.	Killing Joke	Crane Wife
next	Buf. S.F.	Killing Joke	Empire	Stadium Arc.
T+60d	D. of Future P.	Even Now	L. Mks. Wd.	Przts. Author
SAS.	Crane Wife	Empire	H. Fna. Are	You in Rev.
TiSAS.	Crane Wife	Empire	WTE. P. S.	Stadium Arc.

5.6.3 Recommendation Results. Besides the attention visualization, we also present a part of the recommendation results of the same user in Table 5. Additionally, we also show the results of SASRec and TiSASRec, which only leverage sequential patterns. We find that only TGSRec can predict the ground truth item (*Killing Joke*) in top-4 predictions at the time of interests. When time (e.g., T+30d) becomes close to the predicting timestampe 'next' (i.e., T+34d), the ground truth item appears in the top-4 predictions. We can observe that the top predicted items from SASRec are also recommended by TGSRec, though in lower ranks. It again proves that TGSRec can unify sequential patterns and temporal collaborative signals.

6 CONCLUSION

In this paper, we design a new SR model, TGSRec, to unify sequential patterns and temporal collaborative signals. TGSRec is defined upon the proposed CTBG. We apply a temporal kernel to map continuous timestamps on edges to vectors. Then, we introduce the TCT layer, which can infer temporal embeddings of nodes. It samples neighbors and learns attention weights to aggregate both node embeddings and time vectors. In this way, a TCT layer is able to encode both sequential patterns and collaborative signals, as well as reveal temporal effects. Extensive experiments on five real-world datasets demonstrate the effectiveness of TGSRec. TGSRec significantly outperforms existing transformer-based sequential recommendation models. Moreover, the ablation study and detailed analyses verify the efficacy of those components in TGSRec. In conclusion, TGSRec is a better framework to solve the SR problem with temporal information.

7 ACKNOWLEDGMENTS

This work is supported in part by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941. This work is also partially supported by NSF through grant IIS-1763365 and by UC Davis. This work is also funded in part by the National Natural Science Foundation of China Projects No. U1936213

REFERENCES

[1] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[2] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. 108–116.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint* (2018).

[4] Robin Devooght and Hugues Bersini. 2017. Long and short-term recommendations with recurrent neural networks. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. 13–21.

[5] Ziwei Fan, Zhiwei Liu, Lei Zheng Zheng, Shen Wang, and S. Philip Yu. 2021. Modeling Sequences as Distributions with Uncertainty for Sequential Recommendation. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. ACM.

[6] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *TIIS* 5, 4 (2015), 1–19.

[7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR (SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 639–648.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] Wendi Ji, Keqiang Wang, Xiaoling Wang, TingWei Chen, and Alexandra Cristea. 2020. Hybrid Sequential Recommender via Time-aware Attentive Memory Network. *CIKM* (2020).

[12] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.

[13] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*, Yoshua Bengio and Yann LeCun (Eds.).

[14] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[15] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *SIGKDD*. 447–456.

[16] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *SIGKDD*. 1748–1757.

[17] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *KDD*. ACM.

[18] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. 1419–1428.

[19] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*. 322–330.

[20] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and Philip S Yu. 2020. Dynamic Graph Collaborative Filtering. ICDM.

[21] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *WWW*. 951–961.

[22] Ye Liu, Yao Wan, Jianguo Zhang, Wenting Zhao, and S Yu Philip. 2021. Enriching Non-Autoregressive Transformer with Syntactic and Semantic Structures for Neural Machine Translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 1235–1244.

[23] Zhiwei Liu, Ziwei Fan, Yu Wang, and Philip S. Yu. 2021. Augmenting Sequential Recommendation with Pseudo-Prior Items via Reversely Pre-training Transformer. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[24] Zhiwei Liu, Xiaohan Li, Ziwei Fan, Stephen Guo, Kannan Achan, and Philip S. Yu. 2020. Basket Recommendation with Multi-Intent Translation Graph Neural Network. In *2020 IEEE International Conference on Big Data (Big Data)*. 728–737. https://doi.org/10.1109/BigData50022.2020.9377917

[25] Zhiwei Liu, Mengting Wan, Stephen Guo, Kannan Achan, and Philip S Yu. 2020. BasConv: Aggregating Heterogeneous Interactions for Basket Recommendation with Graph Convolutional Neural Network. In *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 64–72.

[26] Zhiwei Liu, Lei Zheng, Jiawei Zhang, Jiayu Han, and Philip S. Yu. 2019. JSCN: Joint Spectral Convolutional Network for Cross Domain Recommendation. *Bigdata* abs/1910.08219 (2019).

[27] Lynn H Loomis. 2013. *Introduction to abstract harmonic analysis*. Courier Corporation.

[28] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled Self-Supervision in Sequential Recommenders. In *SIGKDD*. 483–491.

[29] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.

[30] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *The Web Conference*. 969–976.

[31] Bo Peng, Zhiyun Ren, Srinivasan Parthasarathy, and Xia Ning. 2020. M2: Mixed Models with Preferences, Popularities and Transitions for Next-Basket Recommendation. *arXiv preprint arXiv:2004.01646* (2020).

[32] Bo Peng, Zhiyun Ren, Srinivasan Parthasarathy, and Xia Ning. 2021. HAM: hybrid associations models for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

[33] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys*. 130–137.

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[35] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.

[36] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*. 1441–1450.

[37] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.

[39] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *SIGKDD*. 950–958.

[40] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[41] Yu Wang, Zhiwei Liu, Ziwei Fan, Philip S Yu, and Lichao Sun. 2021. DSKReG: Differentiable Sampling on Knowledge Graph forRecommendation with Relational GNN. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. ACM.

[42] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. 495–503.

[43] Jibang Wu, Renqin Cai, and Hongning Wang. 2020. Déjà vu: A Contextualized Temporal Attention Mechanism for Sequential Recommendation. In *The Web Conference*. 2199–2209.

[44] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential Recommendation Via Personalized Transformer. In *RecSys*. ACM, 328–337.

[45] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*, Vol. 33. 346–353.

[46] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. 2010. Temporal recommendation on graphs via long-and short-term preference fusion. In *KDD*. 723–732.

[47] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*. SIAM, 211–222.

[48] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2019. Self-attention with functional time representation learning. In *NeurIPS*. 15915–15925.

[49] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive Representation Learning on Temporal Graphs. *arXiv preprint arXiv:2002.07962* (2020).

[50] Wenwen Ye, Shuaiqiang Wang, Xu Chen, Xuepeng Wang, Zheng Qin, and Dawei Yin. 2020. Time Matters: Sequential Recommendation with Complex Temporal Information. In *SIGIR*. 1459–1468.

[51] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *SIGIR*. 729–732.

[52] Jian-Guo Zhang, Kazuma Hashimoto, Yao Wan, Ye Liu, Caiming Xiong, and Philip S Yu. 2021. Are Pretrained Transformers Robust in Intent Classification? A Missing Ingredient in Evaluation of Out-of-Scope Intent Detection. *arXiv preprint arXiv:2106.04564* (2021).

[53] Yao Zhang, Yun Xiong, Xiangnan Kong, Zhuang Niu, and Yangyong Zhu. 2019. IGE+: A Framework for Learning Node Embeddings in Interaction Graphs. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[54] Yao Zhang, Yun Xiong, Xiangnan Kong, and Yangyong Zhu. 2017. Learning node embeddings in interaction graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 397–406.

[55] Lei Zheng, Ziwei Fan, Chun-Ta Lu, Jiawei Zhang, and Philip S. Yu. 2019. Gated Spectral Units: Modeling Co-Evolving Patterns for Sequential Recommendation. In *ACM SIGIR*. Association for Computing Machinery, New York, NY, USA, 1077–1080.