Temporal **G**raph **S**equential **Rec**ommender

accepted by 2021 CIKM

code

challenge & motivation.

challenge 1. it is hard to simultaneously encode collaborative signals and sequential patterns.

challenge 2. 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.

summarize ; lacking the consider of between cross-signal

→ notice the importance of **time span**

solution & model architecture

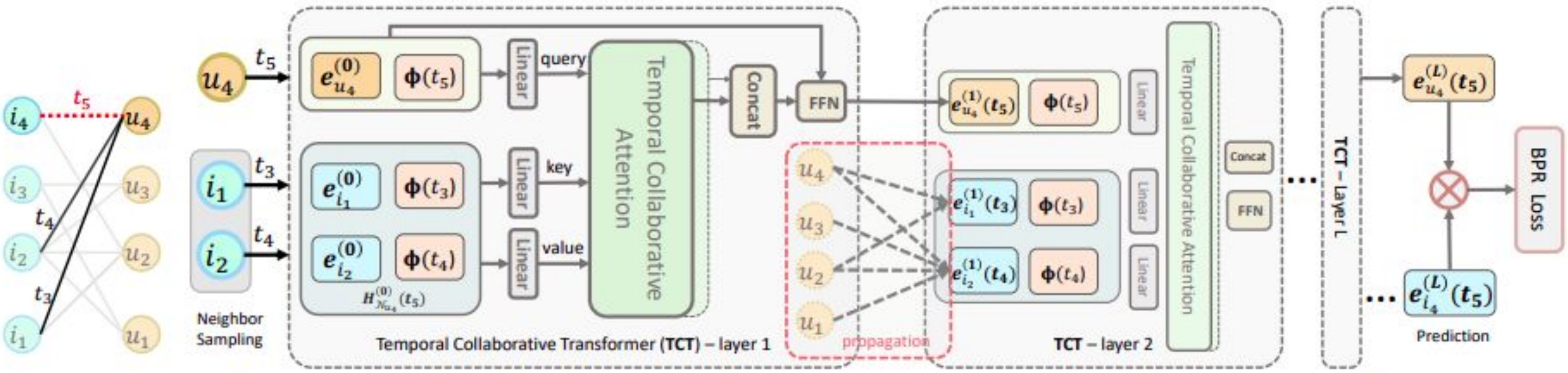


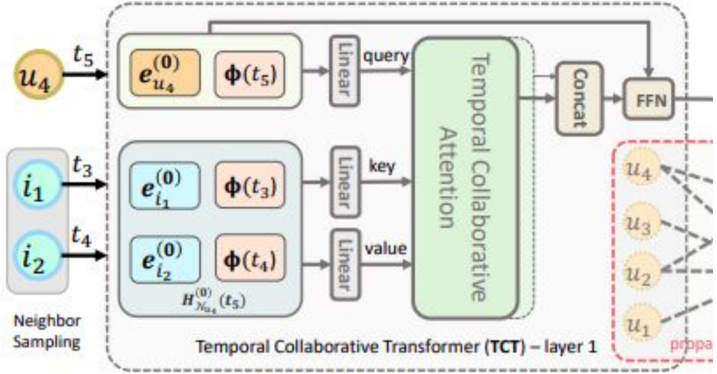
Figure 3: The framework of TGSRec. The query node is u_4, whose final temporal embedding at time t_5 is $h_{u_4}^{(2)}(t_5)$. The TCT layer samples its neighbor nodes and edges. Timestamps on edges are encoded as vectors by using mapping function Φ. Node embeddings for the first TCT layer are long-term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

2 novelty.

1. the Temporal Collaborative Transformer
    - explicitly model collaborative signals in sequences and express temporal correlations of items in sequences.
2. graph information propagation
    - preserve sequential patterns of neighbor items of users.

solution & model architecture

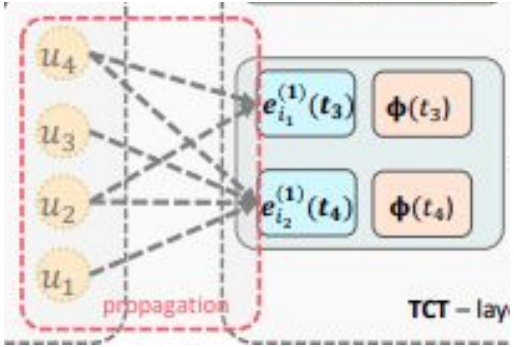
1. the Temporal Collaborative Transformer
2. graph information propagation



- 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.

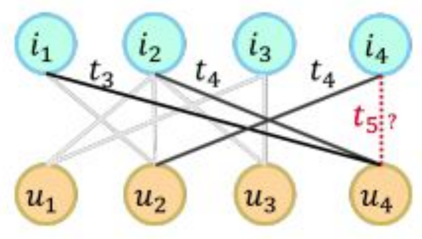
solution & model architecture

1. the Temporal Collaborative Transformer
2. graph information propagation



- ❑ To propagate the information each of ‘user’ or ‘item’ utilize the history information.

Detailed graph construction



assumption ; e-commerce only

Bipartite Graph = {U,I}

→ Continuous-time Bipartite Graph = {U,I, E_T}

where U, I are two disjoint node sets of users and items, respectively.
Every edge e \in E_T is denoted as a tuple e = (u, i ,t), where u \in U , i \in I , and t \in R^+ as the edge attribute.

each triplet (u,i,t) denotes the interaction of a user u with item i at timestamp t.

Embedding Layer

1. Long-term User/Item Embeddings.

→ such as node features and optimized to model the holistic structural information. Long-term embeddings for users and items are necessary for long-term collaborative signals representation.

parameterized by a vector e_u(e_i) from embedding table which is d \times |V| dimensions , V is the sum of count users and items.

2. Continuous-Time Embedding.

→ maps those scalar timestamps into vectors.

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

** temporal effects ; as a function of time span in continuous timespace
for instance, given a pair of interactions (u,i,t_1) , (u,j,t_2) of the same user the temporal effect is defined as a function which is expressed as **kernel value** of the time embeddings of t_1 and t_2 (above notation).
** why kernel value ?
→ temporal effects as a kernel is generalized to any timestamp as it models the time representations directly. Based on Bochner's Theorm(fourier transform).

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

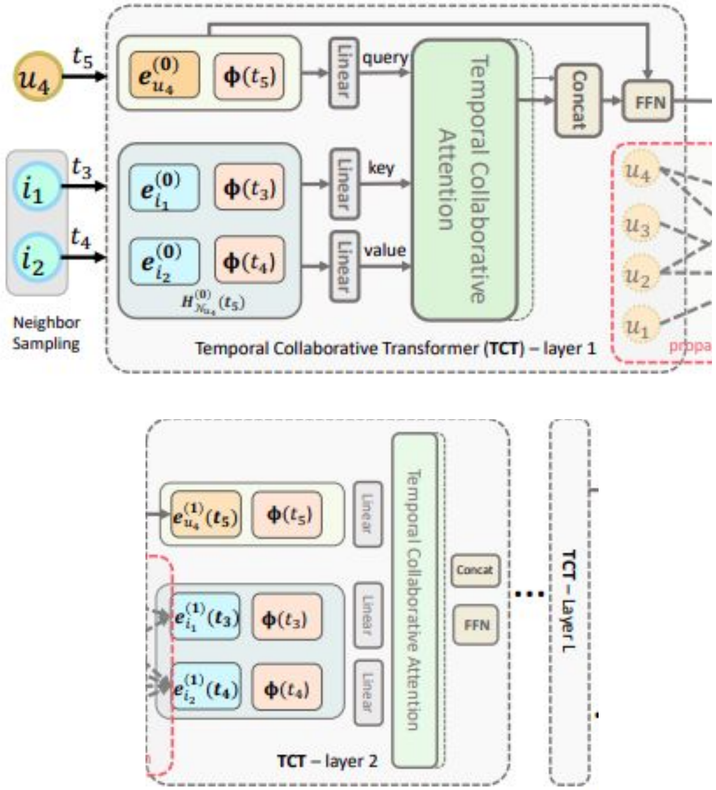
Embedding Layer

```
class PosEncode(torch.nn.Module):
    def __init__(self, expand_dim, seq_len):
        super().__init__()

        self.pos_embeddings = nn.Embedding(num_embeddings=seq_len, embedding_dim=expand_dim)

    def forward(self, ts):
        # ts: [N, L]
        order = ts.argsort()
        ts_emb = self.pos_embeddings(order)
        return ts_emb
```

Continuous-Time Embedding code view

```
class DisentangleTimeEncode(torch.nn.Module):
    def __init__(self, components, expand_dim, factor=5):
        super(DisentangleTimeEncode, self).__init__()

        time_dim = expand_dim
        self.factor = factor
        self.components = components

        init_freq = np.zeros((self.components, time_dim))
        for i in range(self.components):
            #span_range = 2 + int(np.exp(i))
            span_range = 10 + i
            init_freq[i, :] = 1 / span_range ** np.linspace(0, (span_range - 1), time_dim)
        self.basis_freq = torch.nn.Parameter(torch.from_numpy(init_freq).float())
        self.phase = torch.nn.Parameter(torch.zeros(self.components, time_dim).float())

    def forward(self, ts):
        # ts: [N, L]
        batch_size = ts.size(0)
        seq_len = ts.size(1)

        ts = ts.view(batch_size, seq_len, 1, 1) # [N, L, 1, 1]
        map_ts = ts * self.basis_freq.view(1, self.components, -1) #[N, L, components, time_dim]
        map_ts += self.phase.view(1, self.components, -1)

        harmonic = torch.cos(map_ts)

        return harmonic
```

Temporal Collaborative Transformer



two strengths of 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.

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

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.

$$N_u(t) = \{(i, t_s) | (u, i, t_s)$$

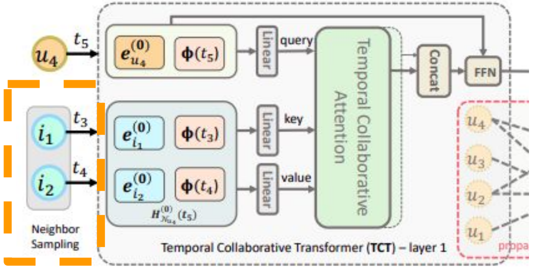
objection ; Gathering the information From user perspective.

(left notation) combination of long term node embeddings(e_u) and time embeddings(\psi(t)).

(right notation) randomly sample S different interactions of u before time t. the query node itself,
to we also propagate temporal collaborative information from its neighbors.

Temporal Collaborative Transformer — Information Propagation

$$e^{(l)}_{N_u}(t) = \sum_{(i,t_s)\in N_u(t)} \pi^u_t(i,t_s)\,W^{(l)}_v\,h^{(l-1)}_i(t_s),$$



objection ; propagate the information of sampled neighbors N_u(t) to infer the temporal embeddings.

- \pi^u_t(i,t_s)^1 denotes the importance of an interaction (u,i,t_s)
- W_v \ in R \in^d \times (d + d_T) is the linear transformation matrix.
- \pi^u_t(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.

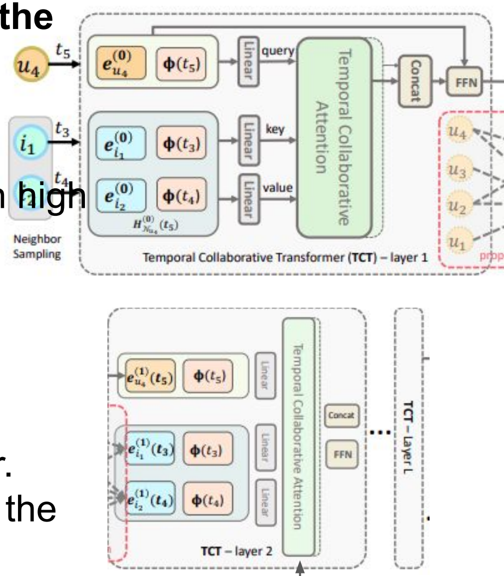
$$\pi_t^u(i,t_s) = \frac{1}{\sqrt{d+d_T}}\left(\boldsymbol{W}_k^{(l)}\boldsymbol{h}_i^{(l-1)}(t_s)\right)^\top \boldsymbol{W}_q^{(l)}\boldsymbol{h}_u^{(l-1)}(t)$$

objection ; to measure the weights \pi^u_t(i,t_s), which considers both neighboring interactions and the temporal information on edges. Both factors contribute to the importance on edges.

- W_k^(l) and W_q^(l) are both linear transformation matrices
- \left(\frac{1}{\sqrt(d+d_T} \right) protects the dot-product from growing large with high dimensions.

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

dot-product attention reason ; we ignore transformation matrices and the scalar factor. the first term denotes the user-item collaborative signal, and the second term models the temporal effect with more stacked layers.



Attention block inputs
(query , key, value)

note $\boldsymbol{K}_u^{(l-1)}(t) = \boldsymbol{W}_k^{(l)}\boldsymbol{H}_{\mathcal{N}_u}^{(l-1)}(t)$, $\boldsymbol{V}_u^{(l-1)}(t) = \boldsymbol{W}_v^{(l)}\boldsymbol{H}_{\mathcal{N}_u}^{(l-1)}(t)$ 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.

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

objection ; aggregate the query information and the neighbor information. so then concatenate and send them to a feed-forward neural network (FFN)

- ❏ e_u^(l)(t) is the temporal embedding of u at t on l-th layer.
- ❏ FFN consists of two linear transformation layers with a ReLU activation function.

** **Generalization to items**. Through we only present the TCT layer from the user query perspective, it is analogous to the query of user , (alternative to user → item.)

output , model prediction and model optimization

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

output (positive)   output (negative   time embedding

- \theta includes long-term embedding E, time embedding parameter w and all linear transformation matrices.

Experiments

: Static

: Temporal

: Transformer-based SR(Sequential Recommendation)

: Others SR(Sequential Recommendation)

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.

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

Non-reproducible the results * 22/03/20

Ablation study

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**

Temporal collaborative attention. (rows 1–2)

Continuous-time embedding. (rows 3–5)

loss function. (row 6)

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

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

- differentiate between brp and bce is whether use 'adaptive fashion' or not