CPMR: Context-Aware Incremental Sequential Recommendation with Pseudo-Multi-Task Learning

Qingtian Bian
Nanyang Technological University
Singapore
bian0027@e.ntu.edu.sg

Jiaxing Xu
Nanyang Technological University
Singapore
jiaxing003@e.ntu.edu.sg

Hui Fang
Shanghai University of Finance and Economics
China
fang.hui@mail.shufe.edu.cn

Yiping Ke
Nanyang Technological University
Singapore
ypke@ntu.edu.sg

ABSTRACT

The motivations of users to make interactions can be divided into static preference and dynamic interest. To accurately model user representations over time, recent studies in sequential recommendation utilize information propagation and evolution to mine from batches of arriving interactions. However, they ignore the fact that people are easily influenced by the recent actions of other users in the contextual scenario, and applying evolution across all historical interactions dilutes the importance of recent ones, thus failing to model the evolution of dynamic interest accurately. To address this issue, we propose a **C**ontext-Aware **P**seudo-**M**ulti-Task **R**ecommender System (CPMR) to model the evolution in both historical and contextual scenarios by creating three representations for each user and item under different dynamics: static embedding, historical temporal states, and contextual temporal states. To dually improve the performance of temporal states evolution and incremental recommendation, we design a Pseudo-Multi-Task Learning (PMTL) paradigm by stacking the incremental single-target recommendations into one multi-target task for joint optimization. Within the PMTL paradigm, CPMR employs a shared-bottom network to conduct the evolution of temporal states across historical and contextual scenarios, as well as the fusion of them at the user-item level. In addition, CPMR incorporates one real tower for incremental predictions, and two pseudo towers dedicated to updating the respective temporal states based on new batches of interactions. Experimental results on four benchmark recommendation datasets show that CPMR consistently outperforms state-of-the-art baselines and achieves significant gains on three of them. The source code is available at https://github.com/DiMarzioBian/CPMR.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Multi-task learning**.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3615512

KEYWORDS

Recommender Systems, Incremental Recommendation, Context-aware Recommendation, Graph Neural Networks, Pseudo-Multi-Task Learning

ACM Reference Format:
Qingtian Bian, Jiaxing Xu, Hui Fang, and Yiping Ke. 2023. CPMR: Context-Aware Incremental Sequential Recommendation with Pseudo-Multi-Task Learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583780.3615512

1 INTRODUCTION

For human beings, making interactions is an important behavior to understand other objects and update their own recognitions. With the time information available, interaction trajectories are typically modeled as chronologically ordered sequences in real applications, e.g., e-commerce (click, add to cart, buy, and even neglect while browsing) [48], music apps (listen for a while or switch rapidly) [3], as well as social media (post, reply, forward, and mention) [8]. The motivations for making interactions vary and generally fall into two types: static preference (long-term interest) and dynamic interest (short-term interest). How to properly model these two types of user interest from interaction sequences breeds the need for effective sequential recommender models.

Different from traditional sequential recommender systems [4, 14, 17, 25] that only make use of the chronological order of interactions, recent studies [10, 16] have proven that timestamps of interactions are more informative in characterizing temporal dynamics. In terms of time modeling, some representation learning works employ additive or concatenative temporal embeddings [10, 16, 40, 41]. Some further model the continuous decay in the effect of interactions after their occurrences [5, 49]. Moreover, to better capture the sequential information from discrete occurrences of interactions in continuous time, hybrid models mixing temporal point processes and recommender systems [2, 7, 27, 28] manage to calculate time-based intensity on items to give recommendations. Some graph-based works [10, 30] also introduce time-based graphs to model complex dynamic connectivities and adopt additive or concatenative temporal embeddings. The above-mentioned studies

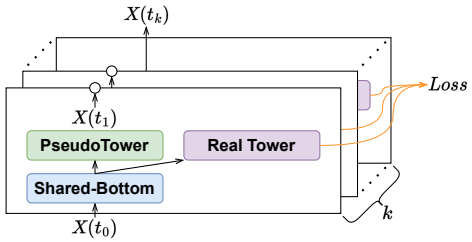

Figure 1: A simple illustration of our proposed Pseudo-Multi-Task Learning paradigm over incremental recommendation task.

model temporal representations in the same way as static embeddings, and thus fail to capture complex dynamics in time-varying interests.

To fully represent the dynamic states embeddings over time, IncCTR [32] introduces incremental learning in Click-Through Rate prediction. SML [44] and FIRE [38] consider the time efficiency and train their models in a fast incremental learning manner without querying historical data. Besides these works, two evolution models for incremental sequential recommendation, JODIE [15] and CoPE [45], have been proposed. JODIE employs a recurrent neural network structure to discretely model the interest trajectories on dynamic embeddings. CoPE approximates the continuous evolution between sets of concurrent interactions by using CGNN [10] to aggregate all historical interactions. Both evolution models learn static embeddings and temporal states, representing their static attributes and dynamic properties as functions of time, respectively.

Despite their superiority over models without considering temporal states, these evolution models evolve their temporal states across the entire history graph without specially treating recent interactions. Consequently, they hardly capture the changing dynamics and ignore the timeliness of temporal states, that is, the impact of recent contexts on users' dynamic interests. Considering the fact that such contexts are sensitive to time, we propose a new evolution model, namely **C**ontext-Aware **P**seudo-**M**ulti-Task **R**ecommender System (CPMR), for fusing the evolution of both historical and contextual scenarios in an MTL-like manner. Specifically, CPMR creates three representations for each user and item under different dynamics: static embedding, historical temporal states, and contextual temporal states. Based on them, CPMR carries out effective information fusion and efficient joint optimization by designing a Pseudo-Multi-Task-Learning (PMTL) paradigm.

MTL is a natural and principled solution to our recommendation problem as our modeling involves multiple tasks for evolving, updating, and fusing different temporal dynamics, all sharing the same set of temporal states. Conventional MTL cannot be directly applied because not all of our tasks generate losses. Therefore, we devise a new pseudo-MTL specifically for our problem. As shown in Figure 1, tasks of our proposed PMTL can be divided into the real task that generates losses and pseudo tasks that generate updated input for the next recurrence.

CPMR is an evolution model implemented under this PMTL paradigm by recurrently evolving the temporal state. In particular, CPMR leverages recommendation as a real task within the prediction module of the real tower, while employing the updating of temporal states as pseudo tasks within the update modules of the simulated towers. In order to model the continuous evolution and the mutual information sharing of historical and contextual temporal states, CPMR also employs two evolution module instantiations and two fusion module instantiations as the shared-bottom networks in PMTL paradigm. More specifically, the update module captures the instant evolution in temporal states from concurrent interactions, while the evolution module focuses on modeling the continuous evolution in temporal states from intervals between batches of interactions. To distinguish evolution within historical and contextual scenarios, CPMR creates two instances of both modules, each dedicated to leveraging the history graph and the context graph, respectively. In particular, the history graph consists of all the interactions that have occurred, and the context graph consists of the interactions that occurred within a fixed-length sliding time window (context window) from the current moment. After evolution, these temporal states are mutually updated at the user-item levels in two fusion module instantiations.

Empowered by the PMTL paradigm and context awareness, CPMR is able to dynamically model the historical and contextual temporal states of users and items and make recommendations. Experimental results demonstrate that CPMR outperforms state-of-the-art baselines and achieves 30.98% gains on MRR and 27.39% gains on Recall@10. Our contributions are summarized as follows:

- We propose a Pseudo-Multi-Task-Learning module to stack single-target incremental recommendations into one multi-target task by mutually evolving temporal states between each task.
- We devise CPMR based on the PMTL paradigm, which enables the evolution and fusion of user interests and item attributes as temporal states within both historical and contextual scenarios.
- We conduct extensive experiments on four datasets to evaluate the effectiveness of CPMR and perform ablation studies on fusion modules and proposed contextual temporal states. The results show that CPMR consistently outperforms state-of-the-art baselines, where both proposed components play important roles.

2 RELATED WORK

Sequential Recommendation (SR): it is a task to predict the next behavior leveraging from the sequence of chronologically ordered historical behaviors. The earliest work, FPMC [20], utilized the Markov Chain to model the transition pattern within sequences. To harness the representation learning capability of deep learning, CNN-based CASER [25] viewed sequential embeddings as images to extract information. Recent sequential recommender works can be divided into two categories: recurrent-based methods [9, 34, 39, 46, 52] and attention-based methods [14, 22, 42, 43, 47, 48].

Some sequential recommender systems explore temporal information to enhance representation learning. For example, [40] conducted extensive experiments on different types of temporal embeddings. TiSASRec [16] embedded relative time intervals to associate with time. CTRec [2] and DeepCoevolve [7] employed temporal point processes to introduce the temporal dynamics into recommendations. JODIE [15] represented temporal states of users

and items by embedding trajectories. However, the update mechanisms for temporal states in these models largely rely on the entire history while ignoring the influence of context. To address this limitation, we calibrate the evolution by exploiting both historical and contextual information in a more timely manner.

Graph-based Recommendation: As each sequence can be viewed as a subgraph, the recommendation dataset can be transformed into a user-item bipartite graph or an item-item graph. For instance, SR-GNN [36] firstly introduced GNN techniques into recommendation tasks. LightGCN [13] designed a simple but effective graph convolution network (GCN). Subsequently, a number of studies apply graph learning to a variety of different recommendation tasks [17, 21, 36, 50], showing the potential of this combination.

Besides, temporal information can also be applied to graph-based recommendations. TGSRec [10] unified sequential patterns and temporal collaborative signals to improve recommendation. CoPE [45] proposed a CGNN-based method to learn from continuous propagation and evolution. FIRE [38] designed graph filters from a graph signal processing perspective to capture the temporal dynamics and address the cold-start problem. RETE [31] proposed a retrieval-enhanced recommendation model based on knowledge graphs to model the temporal dynamics. However, the number of new interactions at each timestamp is too small compared to all historical interactions. Hence applying GNN directly on the graph of all historical interactions cannot effectively capture the dynamics of users and items. On the contrary, in this work, we propose to mitigate this problem by applying another GNN on a more dynamic context graph.

Multi-Task Recommendation: Multi-Task Learning (MTL) is an active research topic in recommender systems, drawing attention from both industry and research. The general network architecture of MTL [1, 18, 19, 23, 24] consists of a shared bottom network that learns task-shared knowledge and multiple task-specific towers to generate the results required by respective tasks. According to the setting of the targets, MTL models can be divided into two categories, one is to use auxiliary tasks to assist in optimizing single or multiple target tasks [29, 35], and the other is to optimize all tasks at the same time [11, 51]. By definition and task settings, common single-task problems do not have to be transformed into multi-task problems. However, if we consider the evolving temporal states as inputs to the shared-bottom network, one tower making recommendations and the other towers updating the input temporal states for further recommendations, we can jointly optimize these single-task recommendations in an MTL-like way by sharing the evolving temporal states among them.

3 PRELIMINARIES

3.1 Problem Formulation

We provide a formal definition of the incremental sequential recommendation (ISR) task we are tackling. Assume we have a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$. We see each user-item interaction with a timestamp as a triplet (u, i, t). Therefore, at a given timestamp t_k, a user $u_p \in \mathcal{U}$ has a chronologically ordered historical interaction sequence $S_p = \{(u_p, i_1, t_1), ..., (u_p, i_{k-1}, t_{k-1})\}$ and $0 = t_0 \leq t_1 \leq t_2 \leq ... \leq t_{k-1}$. If u_p makes an interaction at t_k, ISR aims to predict the ground-truth item i_k that u_p will interact with at t_k by mining u_p's interest from S_p together with all observed interactions from other users before t_k. That is to say, in terms of time, this task strictly adheres to the principle of no data leakage. Unlike some other incremental recommendation tasks [32, 38], the ISR task allows all history data to be used, rather than using only the model's current states and incoming interactions. Based on the definition of ISR, JODIE, CoPE and our CPMR can all be classified as ISR models. The problem formulation of ISR is given as:

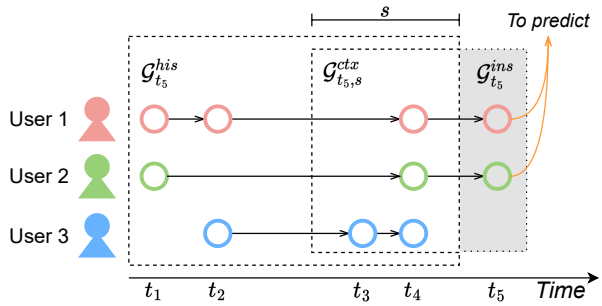

Figure 2: Illustration of the history graph, context graph and instant graph.

Input: The historical interactions of all users before timestamp t_k.
Output: A recommender system that estimates the probability of user u_p interacting with every candidate item $i \in \mathcal{I}$ at t_k, and recommends a top N recommendation list with the highest probabilities to user u_p.

3.2 Graph Formulation

By joining the interactions of all sequences into one edge set $\mathcal{E} = S_1 \cup S_2 \cup ... \cup S_n$, we can represent this edge set as a bipartite interaction graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{I}, \mathcal{E})$, in which $\forall u \in \mathcal{U}$, all its neighbor nodes $\mathcal{N}(u)$ are of item type, i.e., $\mathcal{N}(u) \subseteq \mathcal{I}$, and vice versa. Without loss of generality, we normalize the timestamps in the dataset into $[0, 1]$ and define the moments right before and right after the timestamp t_k as t_k^- and t_k^+. As shown in Figure 2, we further define three types of graphs based on the time spans in CPMR.

Definition 3.1 (Instant Graph). Given an interaction graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{I}, \mathcal{E})$, the instant graph at timestamp t_k is formed by all the interactions happened right at t_k, i.e., $\mathcal{G}^{ins}_{t_k} = \left(\mathcal{U} \cup \mathcal{I}, \mathcal{E}^{ins}_{t_k}\right)$ and $\mathcal{E}^{ins}_{t_k} = \{(u', i', t') \in \mathcal{E} | t' = t_k\}$.

Definition 3.2 (History Graph). Given an interaction graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{I}, \mathcal{E})$, the history graph at timestamp t is formed by all the interactions happened before t, i.e., $\mathcal{G}^{his}_t = \left(\mathcal{U} \cup \mathcal{I}, \mathcal{E}^{his}_t\right)$ and $\mathcal{E}^{his}_t = \{(u', i', t') \in \mathcal{E} | t' < t\}$. Each history graph stays unchanged between two adjacent interactions, i.e., for $\forall t \in (t_{k-1}, t_k)$, $\mathcal{G}^{his}_t = \mathcal{G}^{his}_{t^+_{k-1}} = \mathcal{G}^{his}_{t^-_k}$.

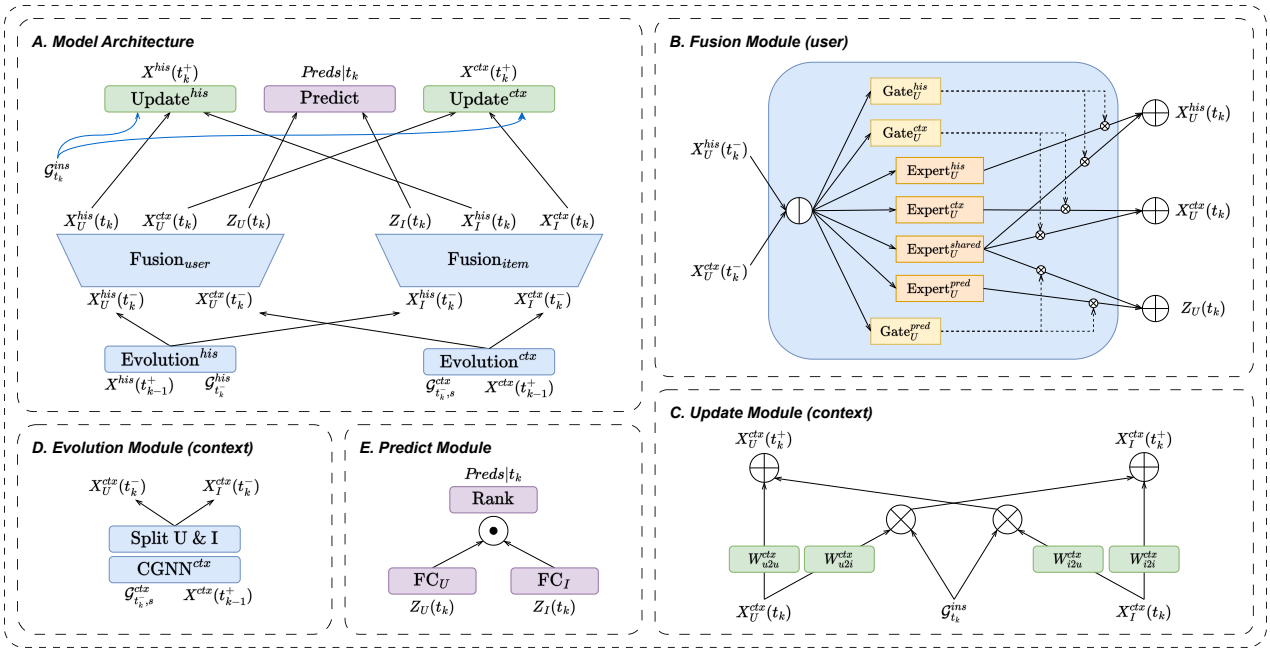

Figure 3: The proposed structure of Context-Aware Pseudo-Multi-Task Recommender System. In subfig A, the shared-bottom network consists of all blue blocks, and each green or pink block represents a pseudo tower or a real tower.

Definition 3.3 (Context Graph). Given an interaction graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{I}, \mathcal{E})$ and the length of context window s, the context graph at timestamp t is formed by all the interactions happened between the interval $[t-s, t)$, i.e., $\mathcal{G}_{t,s}^{ctx} = \left(\mathcal{U} \cup \mathcal{I}, \mathcal{E}_{t,s}^{ctx}\right)$ and $\mathcal{E}_{t,s}^{ctx} = \{(u', i', t') \in \mathcal{E} | t - s \le t' < t\}$, and $\mathcal{E}_{t,s}^{ctx} \subseteq \mathcal{E}_t^{his}$.

Given an interaction graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{I}, \mathcal{E})$, its adjacency matrix is denoted by $\boldsymbol{adj} = \begin{bmatrix} 0 & \boldsymbol{B} \\ \boldsymbol{B}^T & 0 \end{bmatrix}$, where $\boldsymbol{B} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ is the bi-adjacency matrix containing all user-item interactions in the interaction graph that gives $\boldsymbol{B}_{u,i} = \begin{cases} 1 & \text{if } (u,i,t) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$ Because of the sparsity problem in the recommendation dataset, the degrees of nodes can be very different. Therefore we normalize the adjacency matrix and give it as $\widetilde{\boldsymbol{adj}} := \frac{\alpha_0}{2}(\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}} \cdot \boldsymbol{adj} \cdot \boldsymbol{D}^{-\frac{1}{2}})$, where $\boldsymbol{D}$ denotes the degree matrix of $\boldsymbol{adj}$, and α_0 is set to 0.98 to make sure all eigenvalues of $\widetilde{\boldsymbol{adj}}$ to be in the interval $[0, 1)$ for future modeling and approximations [6].

4 METHODOLOGY

4.1 Pseudo-Multi-Task Learning Paradigm

Previous evolution models, JODIE [15] and CoPE [45], recurrently evolve temporal states to make recommendations as they only model a single type of temporal dynamics, i.e., historical dynamics. In this work, we propose to model contextual dynamics in addition to the historical one. By doing so, three different tasks naturally arise: instant update of contextual/historical temporal dynamics, the fusion and continuous evolution of these two types of temporal dynamics, as well as the recommendation task. Beneath all the tasks, they share the same set of user/item temporal states. Meanwhile, each individual task has its own characteristics and target. This motivates us to employ Multi-Task Learning (MTL) as a principled solution. However, each task in conventional MTL generates its own loss, while this is not the case in our problem. Therefore, we design a new pseudo-MTL (PMTL) paradigm to better fit our need in ISR. Specifically, we designate a task that generates losses as a real task, while those not bound with losses as pseudo tasks. Figure 1 illustrates our proposed PMTL paradigm over the task of ISR. The shared-bottom network handles the fusion and evolution of temporal states. The output of shared-bottom network are feed into the pseudo tasks for instant updates and to the real task for prediction. In this way, we achieve a joint optimization akin to MTL.

In this section, we devise CPMR based on the PMTL paradigm, a system modeling both contextual and historical temporal states to solve the ISR task. As shown in Figure 3.A, CPMR consists of four different types of modules, each tackling different tasks. (1) Evolution Module: to perform continuous update of temporal states in intervals between batches of interactions. (2) Fusion Module: to fuse information from different dynamics scenarios at the user-item level. (3) Update Module: to conduct instant update of temporal states based on new batches of interactions. (4) Predict Module: to make incremental recommendations. Following our PMTL paradigm, CPMR instantiates two evolution modules and two fusion modules as the shared-bottom network (blue blocks in Figure 3.A).

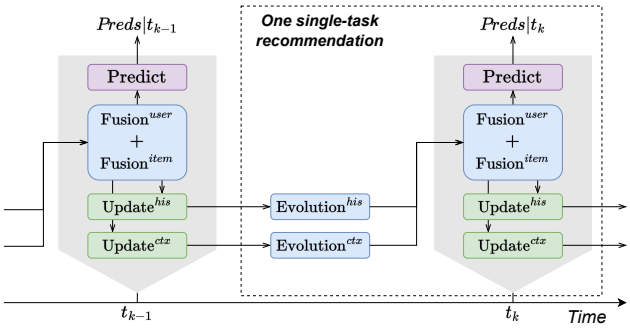

Figure 4: Workflow of the CPMR. Modules in the same grey block are executed in a short time.

With the temporal states generated by the shared-bottom network, CPMR further implements two instantiations of the update module as two pseudo tasks (green blocks) to perform instant updates on temporal states. Finally, CPMR instantiates the real task of PMTL by a predict module (pink block), which generates recommendation losses. While the pseudo tasks do not directly generate losses, its updated temporal states make the next inputs different from the current inputs, and thus the next recurrence's loss will be different. By jointly optimizing these losses from different recurrences, our model can not only be efficiently trained with less time of back-propagation but also reduce overfitting on specific recurrences.

4.2 Overview of CPMR

To be specific, CPMR learns three sets of vector representations for each user and item, static embeddings, historical temporal states and contextual temporal states, which capture the basic preference (attributes) and the time-varying dynamics in historical and contextual scenarios respectively. On top of these representations we implement the aforementioned modules as:

- **Evolution Module** Through two parallel GNN encoders, the historical and contextual temporal states of users and items evolve from t_{k-1}^+ to t_k^-. This module contains two instantiations respectively for historical and contextual scenarios, each incorporating the corresponding input graph. The module is designed to approximate the continuous states evolution at interval (t_{k-1}^+, t_k^-).
- **Fusion Module** The mutual update of temporal states is performed in this module. It takes the latest historical and contextual temporal states at t_k^- as inputs and generates final users' and items' representations for recommendations at t_k, and the updated historical and contextual temporal states at t_k via information fusion. Unlike in the evolution module, the two instantiations in this module are tailored for users and items respectively. They aim to blend the information from both historical and contextual sources and inject such information into the temporal states of each type of nodes, i.e., either user or item nodes.
- **Predict Module** The top-N recommendation list is generated by the module. It takes the final users' and items' representations generated by the fusion module at t_k as inputs.
- **Update Module** The update module represents newly arrived concurrent interactions at t_k as an instant graph and discretely updates the relevant temporal states from t_k to t_k^+. Same as in the evolution module, this module has two instantiations respectively for historical and contextual dynamics.

As shown in Figure 4, CPMR runs the evolution module, the fusion module, and the update module in a loop to keep the temporal states updated. According to the model structure illustrated in Figure 3, for $k = 1, 2, ..., n$, we can write CPMR in a recurrent form, ignoring the notations of users and items:

$$\begin{cases} \boldsymbol{X}^{his}(0^+) = \boldsymbol{X}^{ctx}(0^+) = \boldsymbol{E} \\ \boldsymbol{X}^{his}(t_k^-) = \text{Evolution}^{his}\left(\mathcal{G}_{t_k^-}^{his},\ \boldsymbol{X}^{his}(t_{k-1}^+),\ t_{k-1},\ t_k\right) \\ \boldsymbol{X}^{ctx}(t_k^-) = \text{Evolution}^{ctx}\left(\mathcal{G}_{t_k^-,s}^{ctx},\ \boldsymbol{X}^{ctx}(t_{k-1}^+),\ t_{k-1},\ t_k\right) \\ \boldsymbol{X}^{his}(t_k),\ \boldsymbol{X}^{ctx}(t_k),\ \boldsymbol{Z}(t_k) = \text{Fusion}\left(\boldsymbol{X}^{his}(t_k^-),\ \boldsymbol{X}^{ctx}(t_k^-)\right) \\ \boldsymbol{X}^{his}(t_k^+) = \text{Update}^{his}\left(\mathcal{G}_{t_k}^{ins},\ \boldsymbol{X}^{his}(t_k)\right) \\ \boldsymbol{X}^{ctx}(t_k^+) = \text{Update}^{ctx}\left(\mathcal{G}_{t_k}^{ins},\ \boldsymbol{X}^{ctx}(t_k)\right) \end{cases} \tag{1}$$

where $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{U}\cup\mathcal{I}|\times d}$ denotes the static embeddings for all users and items, $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{U}\cup\mathcal{I}|\times d}$ denotes the users' and items' temporal states, $\boldsymbol{Z} \in \mathbb{R}^{|\mathcal{U}\cup\mathcal{I}|\times d}$ denotes the users' and items' final representations for making recommendations, and the Fusion function here denotes two fusion module instantiations of user or items for simplicity. In our implementation, each expert and gating network is a linear layer.

One example of the execution flow at the recurrence of t_k is shown in Figure 4, and summarized as follows.

(1) Right after some interactions were made at t_{k-1}, CPMR runs the evolution module to update temporal states from $\boldsymbol{X}(t_{k-1}^+)$ to $\boldsymbol{X}(t_k^-)$ at an interval in-between adjacent interactions (t_{k-1}^+, t_k^-).
(2) Right before some interactions are made at t_k, CPMR first calls the fusion module to update evolved temporal states from $\boldsymbol{X}(t_k^-)$ to $\boldsymbol{X}(t_k)$ via information fusion and then calls the predict module to make top-N recommendations for each user involved in the coming interactions $\mathcal{E}_{t_k}^{ins}$.
(3) At timestamp t_k, these interactions are made and the instant graph $\mathcal{G}_{t_k}^{ins} = \left(\mathcal{U}\cup\mathcal{I}, \mathcal{E}_{t_k}^{ins}\right)$ is constructed. If CPMR accumulates losses for a given number of recurrences based on the predictions in Step (2) and the instant graph $\mathcal{G}_{t_k}^{ins}$, it will perform Truncated Back-Propagation Through Time (TBPTT) to update the learnable parameters. Besides that, CPMR also runs the update module to add the effects of this set of interactions $\mathcal{E}_{t_k}^{ins}$ into temporal state $\boldsymbol{X}(t_k)$ to get $\boldsymbol{X}(t_k^+)$. After this, CPMR moves to the next recurrence.

We will introduce each module in detail in subsequent subsections.

4.3 Evolution Module

In the evolution module, the goal is to model new temporal states $\boldsymbol{X}^{his}(t)$ and $\boldsymbol{X}^{ctx}(t)$ for $t \in (t_{k-1}^{+}, t_k^{-})$. Essentially, it models the interest propagation that exists in an interval between adjacent interactions (i.e., an interval without any interactions). Instead of conducting aggregation on all happened interactions ignoring time gaps as what CoPE does, we propose mining from trendy items and their corresponding users by establishing a context environment within the context window and removing the time gaps inside of it. Same to a first-in-first-out queue on the time axis, this context graph can easily adapt to new trends as time goes by.

Specifically, we build our evolution module by employing two parallel CGNNs [37], each of which performs historical dynamics evolution or contextual dynamics evolution respectively. The history graph, formed by all happened interactions, captures the relatively static users' preferences and items' attributes. The context graph, formed by all recent interactions inside one context window, captures dynamic time-varying trends and accommodates potential drifts in users' interests and items' status. In this way, the evolution module in CPMR is able to capture evolution dynamics under both history and context scenarios.

To differentiate the message passing capability of different nodes in an interaction graph, we define the learnable spectral radii by a vector $\boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{U}\cup\mathcal{I}|\times 1}$ for all nodes. Intuitively, each α_i reflects the importance of node i when its embedding is used to form the embeddings of its neighbors. With $\boldsymbol{\alpha}$, we have the learnable adjacency matrix as $\boldsymbol{A} := \text{Broadcast}\left(\text{Sigmoid}(\boldsymbol{\alpha})\right) \odot \widetilde{\boldsymbol{adj}}$ where Broadcast function expands the vector $\boldsymbol{\alpha}$ to $\mathbb{R}^{|\mathcal{U}\cup\mathcal{I}|\times d}$ by copying, $\odot$ denotes element-wise multiplication, and we employ sigmoid function $\text{Sigmoid}(\boldsymbol{\alpha})$ to scale all eigenvalues of $\boldsymbol{A}$ to be in the interval $[0, 1)$, since $\max\left(\text{Sigmoid}(\boldsymbol{\alpha})\right) < 1$. In subsequent discussions, we refer to the learnable adjacency matrix when the adjacency matrix of a history/context/instant graph is used.

Considering that the same user (item) has different interests (attributes) in different scenarios [33], we define the CGNNs' Ordinary Differential Equations (ODEs) for the history graph and context graph respectively as follows:

$$\frac{\mathrm{d}\boldsymbol{X}^{his}(t)}{\mathrm{d}t} = (\boldsymbol{A}_t^{his} - \boldsymbol{I})\boldsymbol{X}^{his}(t) + \boldsymbol{E}, \ \frac{\mathrm{d}\boldsymbol{X}^{ctx}(t)}{\mathrm{d}t} = (\boldsymbol{A}_{t,s}^{ctx} - \boldsymbol{I})\boldsymbol{X}^{ctx}(t) + \boldsymbol{E}, \tag{2}$$

where $\boldsymbol{A}^{his}$ and $\boldsymbol{A}^{ctx}$ use different sets of learnable spectral radii, i.e., $\boldsymbol{\alpha}^{his} \neq \boldsymbol{\alpha}^{ctx}$, to account for the differences between historical and contextual scenarios. With $\boldsymbol{X}^{his}(0) = \boldsymbol{X}^{his}(t_{k-1}^{+})$ and $\boldsymbol{X}^{ctx}(0) = \boldsymbol{X}^{ctx}(t_{k-1}^{+})$, we have the analytical solutions of the two ODEs for $t > t_{k-1}$ and $\Delta t = t - t_{k-1}$ as follows:

$$\begin{cases} \boldsymbol{X}^{his}(t) = \\ \quad (\boldsymbol{A}_t^{his} - \boldsymbol{I})^{-1}\left(e^{(\boldsymbol{A}_t^{his} - \boldsymbol{I})\Delta t} - \boldsymbol{I}\right)\boldsymbol{E} + e^{(\boldsymbol{A}_t^{his} - \boldsymbol{I})\Delta t} \cdot \boldsymbol{X}^{his}(t_{k-1}^{+}), \\ \boldsymbol{X}^{ctx}(t) = \\ \quad (\boldsymbol{A}_{t,s}^{ctx} - \boldsymbol{I})^{-1}\left(e^{(\boldsymbol{A}_{t,s}^{ctx} - \boldsymbol{I})\Delta t} - \boldsymbol{I}\right)\boldsymbol{E} + e^{(\boldsymbol{A}_{t,s}^{ctx} - \boldsymbol{I})\Delta t} \cdot \boldsymbol{X}^{ctx}(t_{k-1}^{+}), \end{cases} \tag{3}$$

where we fix $\boldsymbol{A}_{t,s}^{ctx}$ during the entire interval for computation, i.e., $\boldsymbol{A}_{t,s}^{ctx} = \boldsymbol{A}_{t_k^-,s}^{ctx}$ for $t \in (t_{k-1}, t_k)$. This is considered reasonable as the length of the intervals between adjacent batches of interactions is small compared to the length of the context window. With Eq. (3), we can directly apply approximations of matrix inverse [37] and matrix exponential [45] to obtain a discrete solution.

4.4 Fusion Module

As shown in Figure 3.B, we design the fusion module on top of the Customized Gate Control (CGC) model which is the one-layer version of the well-known Progressive Layered Extraction (PLE) model [24]. In CPMR, we create two fusion module instantiations for users and items separately. Taking the instantiation for users as an example, at $t = t_k$, the inputs are users' historical temporal states $\boldsymbol{X}_U^{his}(t_k^-)$ and contextual temporal states $\boldsymbol{X}_U^{ctx}(t_k^-)$, and the outputs are their updated historical temporal states $\boldsymbol{X}_U^{his}(t_k)$, updated contextual temporal states $\boldsymbol{X}_U^{ctx}(t_k)$, and final representations $\boldsymbol{Z}_U(t_k)$ for recommendation.

To be more specific, we implement one shared expert network, three task-specific expert networks, and three gating networks in Figure 3.B. Each gating network takes the outputs of its corresponding task-specific expert network and the shared expert network to generate task-specific output. By this design, the shared expert network will be affected by all tasks during optimization, but the task-specific expert networks will only be affected by their own tasks. To selectively combine information from shared and task-specific experts, each gating network learns the weights of each expert and sums the experts' output up with these weights. In the instantiation for users, the generation of users' historical temporal states can be summarized as follows:

$$\begin{aligned} \boldsymbol{X}_U^{in} &= \boldsymbol{X}_U^{his}(t_k^-) \oplus \boldsymbol{X}_U^{ctx}(t_k^-), \\ w_U^{his} &= \text{Softmax}\left(\text{Gate}_U^{his}(\boldsymbol{X}_U^{in})\right), \\ \boldsymbol{X}_U^{his}(t_k) &= w_U^{his} \cdot \left[\text{Expert}_U^{his}(\boldsymbol{X}_U^{in}),\ \text{Expert}_U^{shared}(\boldsymbol{X}_U^{in})\right]^T, \end{aligned} \tag{4}$$

where $\oplus$ denotes tensor concatenation on the latent dimension, $\boldsymbol{X}_U^{in}$ denotes concatenated tensor input, w_U^{his} denotes the weights of shared and task-specific expert networks learned from the gating network Gate_U^{his}, Expert_U^{shared} denotes the shared expert network, Expert_U^{his} and Gate_U^{his} denote the expert and gating networks for the task of generating the updated users' historical temporal states. All output terms, including final representations, and historical and contextual temporal states for both users and items, can be generated by the two fusion module instantiations respectively in a similar way.

4.5 Update Module

As evidenced in previous SR models [14, 34], the appearance of a new interaction has an instant impact on its corresponding user and item within a short period of time. Compared to continuous evolution, this process can be seen as discrete. The update module

in our CPMR is designed to implement this sudden change in users' and items' states. At each timestamp $t = t_k$ when some interactions are made, the update module takes the set of new concurrent interactions $\mathcal{E}^{ins}_{t_k}$ as input and transforms the temporal states from $\boldsymbol{X}(t_k)$ to $\boldsymbol{X}(t_k^+)$, which is performed in a discrete way to reflect the instant impact. As shown in Figure 3.C, given the bi-adjacency matrix of the instant graph $\mathcal{G}^{ins}_{t_k}$, $\boldsymbol{B}^{ins}_{t_k}$, the contextual temporal states update procedure for users is formulated as:

$$\begin{aligned} \Delta \boldsymbol{X}^{ctx}_U(t_k) &= \text{ReLU}\left((\boldsymbol{D}^{ins}_{t_k,U})^{-1} \cdot \boldsymbol{W}^{ctx}_{i2u} \cdot \boldsymbol{B}^{ins}_{t_k} \cdot \boldsymbol{X}^{ctx}_I(t_k)\right), \\ \boldsymbol{X}^{ctx}_U(t_k^+) &= \boldsymbol{W}^{ctx}_{u2u} \cdot \boldsymbol{X}^{ctx}_U(t_k) + \boldsymbol{M}_{t_k,U} \odot \Delta \boldsymbol{X}^{ctx}_U(t_k), \end{aligned} \tag{5}$$

where $\boldsymbol{W}^{ctx}_{i2u}$ and $\boldsymbol{W}^{ctx}_{u2u}$ are learnable weight matrices, $\boldsymbol{D}^{ins}_{t_k,U} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{U}|}$ is the diagonal degree matrix on rows of $\boldsymbol{B}^{ins}_{t_k}$, and $\boldsymbol{M}_{t_k,U} \in \{0,1\}^{|\mathcal{U}|\times d}$ is a masking matrix that gives entries of 1 to the users involved in $\mathcal{G}^{ins}_{t_k}$. Other temporal states of users and items can be updated in a similar way.

4.6 Predict Module and Optimization

To provide a top-N recommendation list for each user who makes interactions at $t = t_k$, we calculate the inner-product similarity λ of each user to all items. Given a user u (an item i) at $t = t_k$, we fuse its static embedding $\boldsymbol{E}_u$ ($\boldsymbol{E}_i$) and its final representation $\boldsymbol{Z}_u(t_k)$ ($\boldsymbol{Z}_i(t_k)$) with one linear layer, and compute user-item similarity by inner product:

$$\lambda(u,i,t_k) = \text{FC}_U\left(\boldsymbol{E}_u \oplus \boldsymbol{Z}_u(t_k)\right) \cdot \text{FC}_I\left(\boldsymbol{E}_i \oplus \boldsymbol{Z}_i(t_k)\right), \tag{6}$$

where $\oplus$ denotes concatenation on the latent dimension, FC_U and FC_I are two fully-connected layers.

Optimization During the model optimization, TBPTT is performed every 20 batches (i.e., 20 unique timestamps). For each interaction (u,i,t_k) in $\mathcal{E}^{ins}_{t_k}$, we randomly sample N_{neg} negative users that have never interacted with the item i before t_k, and N_{neg} negative items that user u has never interacted with before t_k. We then create a negative edge set $\mathcal{E}^{neg}_{t_k}$ by connecting sampled negative users to i, and sampled negative items to u. We compute the prediction loss for each interaction $(u,i,t_k) \in \mathcal{E}^{ins}_{t_k}$ via InfoNCE [26]:

$$\mathcal{L}(u,i,t_k) = -\log\left(\frac{e^{\lambda(u,i,t_k)}}{e^{\lambda(u,i,t_k)} + \sum_{(u',i',t_k)\in\mathcal{E}^{neg}_{t_k}} e^{\lambda(u',i',t_k)}}\right). \tag{7}$$

The incremental recommendation loss at $t = t_k$ is computed as the average loss over all concurrent interactions in $\mathcal{G}^{ins}_{t_k}$:

$$\mathcal{L}_{t_k} = \frac{1}{\left|\mathcal{E}^{ins}_{t_k}\right|} \sum_{(u,i,t_k)\in\mathcal{E}^{ins}_{t_k}} \mathcal{L}(u,i,t_k). \tag{8}$$

5 EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness of CPMR. We aim to answer the following questions.

RQ1: How good is the performance of CPMR when compared with state-of-the-art evolution models?

RQ2: How does the length of the context window affect the performance of CPMR?

RQ3: How does the proposed PMTL paradigm affect joint optimization and the mutual update of temporal states?

RQ4: How do the proposed contextual temporal states affect the performance of CPMR compared to historical temporal states?

Table 1: Statistics of Sequential Recommendation Datasets.

	Garden	Video	Games	ML-100K
# Users	1,686	5,130	24,303	943
# Items	962	1,685	10,672	1,349
# Interactions	13,272	37,126	231,780	99,287
# Timestamps	1,888	1,946	5,302	49,119
Span (Days)	5,221	4,984	5,395	214.83

Table 2: Hyperparameters in proposed CPMR.

Hyperparameter	Value
Embedding dimension d	128
# negative users / items N_{neg}	8
Optimizer	Adam
# batches per TBPTT n_{tbptt}	{10, 20, 30, 40}
Learning rate	{5., 2., 1.} × {1e-2, 1e-3, 1e-4, 1e-5}
Weight decay	{5., 2., 1.} × {1e-2, 1e-3, 1e-4, 1e-5}
Learning rate decay	× {0.5, 0.2} every {6, 10} epochs
Context window length s	Search from 5 to 100 at a step of 5

5.1 Experimental Setup

Datasets: We conduct experiments on four public sequential recommendation datasets, including three subsets from Amazon review datasets [12] ('Patio, Lawn and Garden' as Garden, 'Amazon Instant Video' as Video, and 'Video Games' as Games)[1], and one movie rating dataset (ML-100K)[2]. The reason for not using session-based recommendation datasets is that long-term sequences of user interactions can show clearer interest evolution compared to short-term anonymous sessions, and session-based recommendation models usually do not explicitly embed users. We use the same preprocessing pipeline of Caser [25] and CoPE [45], which recursively discards users and items with less than 5 observations until each remaining user or item contains at least 5 interactions. By following CoPE [45] and JODIE [15], each dataset is split by time into 80%/10%/10% as training, validation and test sets, and the timestamps are coarsed into days for efficiency. The statistics of these datasets are summarized in Table 1. We run CPMR and CoPE (ours) five times with different seeds on each dataset to obtain the experimental results.

Baselines: Regarding the baseline selection and quality metrics used, we follow those in CoPE [45]. Specifically, we compare CPMR with various baselines, including (1) graph-based recommendation model, LightGCN [13]; (2) deep recurrent recommendation models, such as Time-LSTM [52] and RRN [34]; (3) temporal network embedding models, such as DeepCoevolve [7], JODIE [15] and CoPE

[1] http://jmcauley.ucsd.edu/data/amazon/index.html
[2] https://grouplens.org/datasets/movielens/

Table 3: Recommendation performance. R@10 is short for Recall@10. The results of all baselines in the first section are imported from CoPE [45]. The results of CoPE (ours) in the second section are derived from our rewritten code. The best results and the runner-up among baselines and CPMR are highlighted in bold and underline respectively. The % Gains are calculated by comparing the best-performing baseline with CPMR. Statistical significance of pairwise differences of CMPR vs. the best baseline is determined by a paired t-test ($***$, $**$ for p-value $\leq 0.01, 0.05$ respectively).

	Garden		Video		Game		ML-100K	
	MRR	R@10	MRR	R@10	MRR	R@10	MRR	R@10
LightGCN	0.025	0.087	0.019	0.036	0.015	0.026	0.012	0.025
Time-LSTM	0.038	0.134	0.028	0.044	0.014	0.020	0.022	0.058
RRN	0.072	0.152	0.033	0.068	0.018	0.029	0.032	0.065
DeepCoevolve	0.046	0.121	0.023	0.050	0.013	0.027	0.029	0.069
JODIE*	0.049	0.127	0.037	0.078	0.021	0.035	0.034	0.074
CoPE*	0.081	0.192	0.048	0.088	0.026	0.047	0.038	0.081
CoPE (ours)	0.0844 ± 0.0012	0.1953 ± 0.0029	0.0421 ± 0.0010	0.0922 ± 0.0014	0.0302 ± 0.0005	0.0585 ± 0.0013	0.0443 ± 0.0008	0.0954 ± 0.0020
CPMR	**0.0853** ± 0.0009	**0.2021** ± 0.0034	**0.0664** ± 0.0019	**0.1327** ± 0.0022	**0.0445** ± 0.0022	**0.0842** ± 0.0034	**0.0522** ± 0.0023	**0.1128** ± 0.0073
% Gain	1.06%	3.48%**	57.71%***	43.92%***	47.35%***	43.93%***	17.83%***	18.23%***

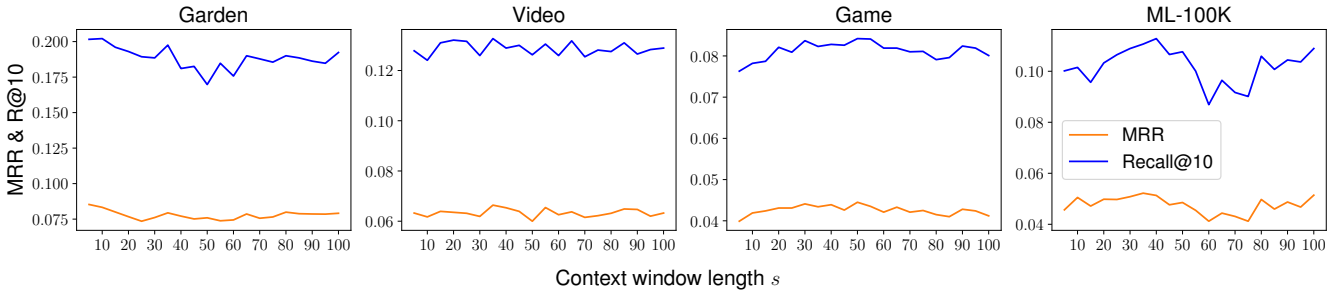

Figure 5: Results on Garden, Video, Game and ML-100K w.r.t. different context window length s (in days).

[45]. For JODIE and CoPE, we use the reported model in the paper of CoPE [45]: JODIE* that disables test-time training and CoPE* that disables test-time training, meta-learning, and jump loss. We also report our own rewritten and re-tuned CoPE (ours) with meta-learning and jump loss implemented. For quality metrics, we use mean reciprocal rank (MRR) for the targets among all items, and Recall@10 scores for target items among top-10 recommendations. For a fair comparison, we use the reported results from CoPE [45] for all baselines in the first section in Table 3.

Hyperparameters: Our selections of hyperparameters are reported in Table 2. We use $d = 128$ as the dimensions of static embeddings and temporal states for CPMR and CoPE (ours). Considering the fact that incremental recommendation is very prone to overfitting, we carefully tune these models on learning rate and weight decay using small step sizes. To figure out how context affects model performance, we also tune the window length from 5 days to 100 days at a step of 5 days.

5.2 Recommendation Performance (RQ1)

As shown in Table 3, CPMR outperforms all baselines on both MRR and Recall@10 across all four sequential recommendation datasets for ISR tasks. Compared with the best-performing baseline models, CPMR achieves 30.98% gains on MRR and 27.39% gains on Recall@10 on average. Statistics significance from paired t-value tests show that CPMR outperforms CoPE significantly on Video, Game and ML-100K datasets where the contextual trends are much more obvious. For Garden, we will explain why the gain is subtle in subsequent sections. The superior performance of CPMR over CoPE is credited to the design of the PMTL paradigm and context awareness. Other temporal baselines perform poorer than CoPE and CPMR because of their recurrent treatment on concurrent interactions. Without time awareness, LightGCN lacks the capability of capturing embedding dynamics, resulting in its inferiority in comparison.

5.3 Length of Context Window (RQ2)

The length of the context window s is an important hyperparameter affecting the context awareness of CPMR. Figure 5 reports the results of this sensitivity study on four datasets. We use context window lengths s in multiples of 5, ranging from 5 to 100. The optimal length will be discussed below.

Garden: The optimal s is 5 days, but the total time span is 5221 days. Such a short context length suggests that hardly any contextual trends exist. Considering that garden tools iterate very slowly, a shorter s matches the reality. The lack of context also results in the smallest improvement ratio on Garden across all datasets.

Table 4: Ablation studies of the fusion module and contextual/historical temporal states.

	Garden		Video	
	MRR	R@10	MRR	R@10
CPMR	**0.0853**$_{\pm 0.0009}$	**0.2021**$_{\pm 0.0034}$	**0.0664**$_{\pm 0.0019}$	**0.1327**$_{\pm 0.0022}$
w/o ctx	0.0789$_{\pm 0.0020}$	0.1935$_{\pm 0.0060}$	0.0565$_{\pm 0.0025}$	0.1078$_{\pm 0.0035}$
w/o his	0.0804$_{\pm 0.0017}$	0.1950$_{\pm 0.0026}$	0.0613$_{\pm 0.0013}$	0.1262$_{\pm 0.0015}$
w/o fusion	0.0838$_{\pm 0.0048}$	0.2021$_{\pm 0.0045}$	0.0611$_{\pm 0.0008}$	0.1171$_{\pm 0.0024}$

ML-100K & Video: The optimal s is 35 days. Since both datasets contain movies and TV series, the length is roughly in line with their showtime and popularity cycle on social networks as well. **Game:** The optimal s is 50 days. Typically, gaming communities can engage in prolonged discussions about popular games, while meticulously crafted games often demand tens to hundreds of hours for completion. In this case, it makes sense to use a longer contextual window.

5.4 Ablation: Context and History (RQ3)

To validate the effectiveness of contextual awareness, we choose the two datasets with the largest and smallest CPMR improvements, Garden and Video, and design two variants for the ablation studies as follows. **w/o ctx:** disable contextual temporal states by removing the contextual instantiation of the evolution module and changing the input of the fusion module into historical temporal states only (including removing tensor concatenation, Gate^{ctx} network and Expert^{ctx} network). **w/o his:** disable historical temporal states by removing the historical instantiation of the evolution module and changing the input of the fusion module into contextual temporal states only (including removing tensor concatenation, Gate^{his} network and Expert^{his} network).

From the second section of Table 4, removing either of the two temporal states leads to a decrease in model performance. This is because one single temporal state contains less information on the dynamics and cannot conduct information fusion in the fusion module. Furthermore, for both datasets, CPMR without history performs better than the version without context, which shows the importance of context awareness when modeling evolution dynamics.

5.5 Ablation: Fusion Module (RQ4)

To validate the effectiveness of the fusion module, we design a variant for this ablation study. **w/o fusion:** remove the fusion module, directly adding involved temporal states $Z(t_k) = X(t_k) = X^{his}(t_k^-) + X^{ctx}(t_k^-)$ as input of update module and predict module.

We can observe from Table 4 that, without the fusion module, the performance of CPMR drops dramatically on Video, but only slightly on Garden. Based on the experiment in Section 5.3, we believe this is because the context of the garden contains too little unique information, which makes the information fusion less effective.

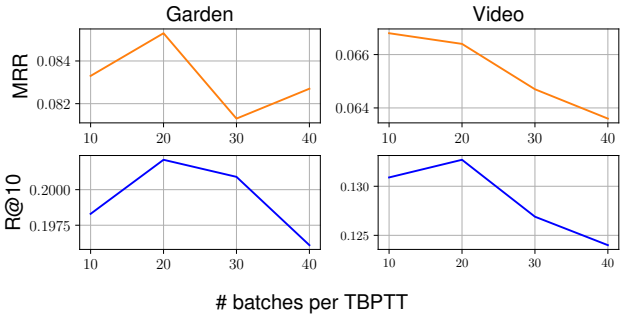

Figure 6: Ablation study on # batches per TBPTT.

To better inspect the performance of the fusion module, we also tune the number of batches per TBPTT, n_{tbptt}, as they can be seen as the number of tasks in MTL. The results in Figure 6 show that choosing an appropriate number is important, as fewer batches may lead to inefficiency and task-specific overfitting, while more batches may also lead to a lack of guidance on temporal state evolution. We observe that setting $n_{tbptt} = 20$ is a good choice for both Garden and Video datasets.

6 CONCLUSION

In this paper, we propose a new Pseudo-Multi-Task Learning paradigm and design a context graph that captures contextual dynamics, distinct from historical dynamics. Based on these, we design a novel recommender system, CPMR. By jointly optimizing multi-target incremental sequential recommendations, CPMR is able to effectively capture and fuse historical and contextual temporal states and outperforms SOTA models by 30.98% on MRR and 27.39% on Recall@10 on average, as demonstrated by extensive experiments. Ablation studies also show the effectiveness of the two proposed designs. Since the great potential of context has been revealed, in the future, we plan to enrich context with other information (e.g., social links with timestamps, cross-domain sequences with timestamps).

7 ACKNOWLEDGMENTS

This research/project is supported by the Ministry of Education, Singapore under its MOE Academic Research Fund Tier 2 (STEM RIE2025 Award MOE-T2EP20220-0006), and the National Research Foundation, Singapore under its Industry Alignment Fund – Prepositioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore, and National Research Foundation, Singapore. This work is also partially supported by Shanghai Rising-Star Program (Grant No. 23QA1403100), the Natural Science Foundation of Shanghai (Grant No. 21ZR1421900), and the National Natural Science Foundation of China (Grant No. 72192832) awarded to Dr. Hui Fang.

REFERENCES

[1] Ting Bai, Yudong Xiao, Bin Wu, Guojun Yang, Hongyong Yu, and Jian-Yun Nie. 2022. A Contrastive Sharing Model for Multi-Task Recommendation. In *Proceedings of the ACM Web Conference 2022*. 3239–3247.

[2] Ting Bai, Lixin Zou, Wayne Xin Zhao, Pan Du, Weidong Liu, Jian-Yun Nie, and Ji-Rong Wen. 2019. CTrec: A long-short demands evolution model for continuous-time recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 675–684.

[3] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The music streaming sessions dataset. In *The World Wide Web Conference*. 2594–2600.

[4] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 378–387.

[5] Sung Min Cho, Eunhyeok Park, and Sungjoo Yoo. 2020. MEANTIME: Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 515–520.

[6] Fan RK Chung. 1997. *Spectral graph theory*. Vol. 92. American Mathematical Soc.

[7] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).

[8] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. 2013. The anatomy of a scientific rumor. *Scientific reports* 3, 1 (2013), 1–9.

[9] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *Proceedings of the eleventh ACM conference on recommender systems*. 152–160.

[10] 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 & Knowledge Management*. 433–442.

[11] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural multi-task recommendation from multi-behavior data. In *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, 1554–1557.

[12] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[15] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1269–1278.

[16] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.

[17] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2020. Memory augmented graph neural networks for sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 5045–5052.

[18] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.

[19] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1137–1140.

[20] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[21] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM international conference on web search and data mining*. 555–563.

[22] 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 *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[23] Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. Learning sparse sharing architectures for multiple tasks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 8936–8943.

[24] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 269–278.

[25] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.

[26] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018). arXiv:1807.03748 http://arxiv.org/abs/1807.03748

[27] Bjørnar Vassøy, Massimiliano Ruocco, Eliezer de Souza da Silva, and Erlend Aune. 2019. Time is of the essence: a joint hierarchical rnn and point process model for time and item predictions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 591–599.

[28] Dongjing Wang, Xin Zhang, Zhengzhe Xiang, Dongjin Yu, Guandong Xu, and Shuiguang Deng. 2021. Sequential Recommendation Based on Multivariate Hawkes Process Embedding With Attention. *IEEE transactions on cybernetics* (2021).

[29] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *The world wide web conference*. 2000–2010.

[30] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 1101–1110.

[31] Ruijie Wang, Zheng Li, Danqing Zhang, Qingyu Yin, Tong Zhao, Bing Yin, and Tarek Abdelzaher. 2022. RETE: Retrieval-Enhanced Temporal Event Forecasting on Unified Query Product Evolutionary Graph. In *Proceedings of the ACM Web Conference 2022*. 462–472.

[32] Yichao Wang, Huifeng Guo, Ruiming Tang, Zhirong Liu, and Xiuqiang He. 2020. A practical incremental method to train deep ctr models. *arXiv preprint arXiv:2009.02147* (2020).

[33] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 169–178.

[34] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 495–503.

[35] Qitian Wu, Lei Jiang, Xiaofeng Gao, Xiaochun Yang, and Guihai Chen. 2019. Feature Evolution Based Multi-Task Learning for Collaborative Filtering with Social Trust.. In *IJCAI*. 3877–3883.

[36] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 346–353.

[37] Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. 2020. Continuous graph neural networks. In *International Conference on Machine Learning*. PMLR, 10432–10441.

[38] Jiafeng Xia, Dongsheng Li, Hansu Gu, Jiahao Liu, Tun Lu, and Ning Gu. 2022. FIRE: Fast Incremental Recommendation with Graph Signal Processing. In *Proceedings of the ACM Web Conference 2022*. 2360–2369.

[39] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S Sheng S. Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent convolutional neural network for sequential recommendation. In *The world wide web conference*. 3398–3404.

[40] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2019. Self-attention with functional time representation learning. *Advances in neural information processing systems* 32 (2019).

[41] Zhen Yang, Ming Ding, Bin Xu, Hongxia Yang, and Jie Tang. 2022. STAM: A Spatiotemporal Aggregation Method for Graph Neural Network-based Recommendation. In *Proceedings of the ACM Web Conference 2022*. 3217–3228.

[42] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential recommender system based on hierarchical attention network. In *IJCAI International Joint Conference on Artificial Intelligence*.

[43] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation.. In *IJCAI*. 4320–4326.

[44] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to retrain recommender system? A sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1479–1488.

[45] Yao Zhang, Yun Xiong, Dongsheng Li, Caihua Shan, Kan Ren, and Yangyong Zhu. 2021. CoPE: Modeling Continuous Propagation and Evolution on Interaction Graph. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2627–2636.

[46] Yu Zheng, Chen Gao, Jianxin Chang, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2022. Disentangling Long and Short-Term Interests for Recommendation. In *Proceedings of the ACM Web Conference 2022*. 2256–2267.

[47] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5941–5948.

[48] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.

[49] Huachi Zhou, Qiaoyu Tan, Xiao Huang, Kaixiong Zhou, and Xiaoling Wang. 2021. Temporal augmented graph neural networks for session-based recommendations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1798–1802.

[50] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. 2020. Improving conversational recommender systems via knowledge graph based semantic fusion. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1006–1014.

[51] Feng Zhu, Chaochao Chen, Yan Wang, Guanfeng Liu, and Xiaolin Zheng. 2019. Dtcdr: A framework for dual-target cross-domain recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1533–1542.

[52] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM.. In *IJCAI*, Vol. 17. 3602–3608.