Leaping Through Time with Gradient-based Adaptation for Recommendation

Nuttapong Chairatanakul[1,2], Hoang NT[1], Xin Liu[3,2,4], Tsuyoshi Murata[1,2]

[1]Tokyo Institute of Technology [2]RWBC-OIL, AIST [3]AIRC, AIST [4]DigiARC, AIST
nuttapong.c@net.c.titech.ac.jp, hoangnt@net.c.titech.ac.jp, xin.liu@aist.go.jp, murata@c.titech.ac.jp

Abstract

Modern recommender systems are required to adapt to the change in user preferences and item popularity. Such a problem is known as the temporal dynamics problem, and it is one of the main challenges in recommender system modeling. Different from the popular recurrent modeling approach, we propose a new solution named LeapRec to the temporal dynamic problem by using trajectory-based meta-learning to model time dependencies. LeapRec characterizes temporal dynamics by two complement components named global time leap (GTL) and ordered time leap (OTL). By design, GTL learns long-term patterns by finding the shortest learning path across unordered temporal data. Cooperatively, OTL learns short-term patterns by considering the sequential nature of the temporal data. Our experimental results show that LeapRec consistently outperforms the state-of-the-art methods on several datasets and recommendation metrics. Furthermore, we provide an empirical study of the interaction between GTL and OTL, showing the effects of long- and short-term modeling.

1 Introduction

Recommender systems (Schafer, Konstan, and Riedl 1999; Ricci, Rokach, and Shapira 2011; Aggarwal 2016) have become essential tools in most real-world applications of big data. These systems ease the users' effort in selecting suitable *items* for their needs (e.g., commercial products, books, services) by finding common patterns of their preferences. In practice, the preferences of users and their opinions of items frequently change over time (Koren 2009; Xiong et al. 2010). It poses a challenge in designing recommender systems: *How to incorporate temporal dynamics into recommendation results?*

Following the footsteps of sequential neural modeling, recommender systems researchers have designed a wide range of recurrent neural networks (RNNs) to solve the temporal dynamic challenge (Hidasi et al. 2016; Li et al. 2017; Zhu et al. 2017; Ma et al. 2020). The main idea of these architectures is to capture the relationship between the current and the past interactions using recurrent units (Wang et al. 2019a). However, applying RNNs to new types of learning models is often difficult (Pascanu, Mikolov, and

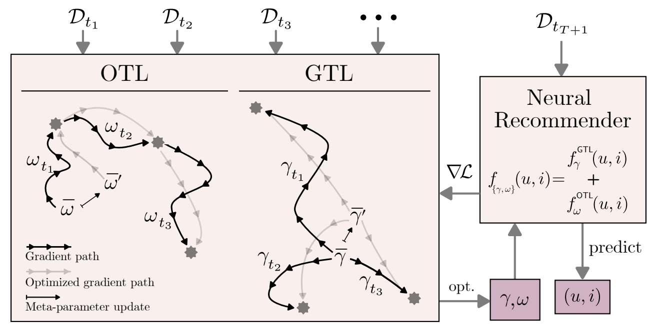


Figure 1: *Left*: Our proposed LeapRec consists of GTL and OTL. *Right*: A Neural Recommender. LeapRec enables any neural recommender to learn temporal dynamics by optimizing the gradient paths $\{\gamma_i\}_{i=t_1}^{t_T}$ and $\{\omega_j\}_{j=t_1}^{t_T}$.

Bengio 2013; Yu et al. 2019a); For example, the graph neural network (GNN) is a promising model that has recently become the state-of-the-art for recommendations on non-temporal datasets (Wang et al. 2019b; He et al. 2020). Applying the recurrent architecture to a GNN would require redesigning the whole neural network (Xu et al. 2019; Rossi et al. 2020). Therefore, an interesting research question arises: *Is there a more versatile alternative solution than RNNs to model temporal dynamics?*

Drawing inspirations from the meta-learning literature, which studies fast adaptation to new tasks (Vilalta and Drissi 2002), we hypothesize that meta-learning methods could be useful to model temporal dynamics. Research on meta-learning for differentiable models demonstrated the advantages of a gradient-based method named MAML (Finn, Abbeel, and Levine 2017) over RNN-based models (Ravi and Larochelle 2016; Santoro et al. 2016) in the few-shot learning problem. This technique leverages the gradient path of the optimization process to learn an initialization of parameters, which allows fast adaptation to unseen tasks.

In this paper, we study the application of meta-learning to model the temporal dynamics in recommender systems. The overall approach is to consider the recommendation problem on data from each time period as a separate task. This approach implies that each time step is modeled as a sequence of gradient steps (a gradient path) in the optimization process. However, a straightforward application of MAML poses two main issues. First, since the computational cost of MAML mainly depends on the number of gradient steps and the number of gradient steps grows linearly with the number of time steps, it becomes infeasible to keep track of long-term time dependency. Second, although we can reduce the computational cost using the first-order approximation of MAML (FOMAML) (Finn, Abbeel, and Levine 2017; Nichol, Achiam, and Schulman 2018), this only considers the start and end points of the gradient paths. That is, FOMAML ignores the *gradient trajectory*[1], which leads to a loss in temporal information. More recently, trajectory-aware meta-learning was proposed by Flennerhag et al. (2019a) to reduce the computational cost of MAML without sacrificing performance by minimizing the total length of gradient paths across tasks. In the temporal recommendation context, we found that Leap is a promising approach because it considers the entire gradient trajectory while maintaining the computation at minimal cost.

We propose LeapRec as a novel adaptation of trajectory-based meta-learning to recommender systems. Figure 1 illustrates an overview of our approach. In LeapRec, we divide temporal dynamics into two categories: global time leap (GTL) and ordered time leap (OTL). GTL finds common patterns that share across all time periods. Examples of these patterns in e-commerce are classic books, basic goods, and all-time best seller items. On the other hand, OTL captures the temporal dynamics from momentary social trends. Our experimental results show that our implementation of LeapRec recommendation model (Figure 2) outperforms the state-of-the-art (SOTA) methods on benchmark datasets by a large margin. Furthermore, LeapRec, when used as a plug-in method for other recommender systems, also significantly improves their performances. We summarize our contributions as follows.

- To the extent of our knowledge, LeapRec is the first trajectory-based meta-learning approach to recommender systems.
- To fully utilize the trajectory-based approach to recommender systems, we propose two components of LeapRec named Global and Ordered Time Leap.
- Our empirical results show a clear improvement over existing approaches, and our empirical analyses explain the dynamic behavior of LeapRec.

2 Related Work

Sequential Recommendation

Sequential history of user interactions provides valuable information about their preferences and dependencies between items. Naturally, RNN-based models (Hidasi et al. 2016; Li et al. 2017; Wu et al. 2017; Chairatanakul, Murata, and Liu 2019) have become the most popular choice for modern recommender systems, owing to their effectiveness in handling sequential data. However, this approach is limited by the memory of RNNs: the amount of memory required to capture long-term dependencies grows linearly with the number of time steps. Since it is infeasible to design such a learning model, Chen et al. (2018); Liu et al. (2018) proposed the use of memory networks (Weston, Chopra, and Bordes 2015) to improve long-term dependencies modeling in recommender systems. In parallel with these developments in RNNs, self-attention (SA) models (Vaswani et al. 2017) have gained popularity due to their success in natural language processing. Their popularity inspires researchers to investigate the applications of self-attention to improve recommendation quality (Kang and McAuley 2018; Sun et al. 2019). These models have the advantage that they can partially adapt to recent data based on the change in input sequences. However, the dependencies between items can be outdated (e.g., combinations of fashion items), and the models cannot differentiate the outdated and new trends.

Temporal-aware Recommendation

In the real world, the preference of users and their opinions of items change over time. Temporal-aware recommendation models explicitly incorporate time into their recommendation results. TimeSVD++ (Koren 2009) is a pioneer work, which includes a time-bias factor. Xiong et al. (2010) proposed tensor factorization with time as a dimension. Along with recent developments in the sequential recommendation, Time-LSTM (Zhu et al. 2017), MTAM (Ji et al. 2020), TiSASRec (Li, Wang, and McAuley 2020), and TGSRec (Fan et al. 2021) incorporate time intervals between successive interactions into LSTM, Memory-Network, and SA; SLiRec (Yu et al. 2019b) and TASER (Ye et al. 2020) consider both time intervals and the time of the prediction. Each design has its own modifications that refine a few specific components to be time-aware. Thus, it is difficult to transfer the design of temporal components to other architectures.

Another type of temporal-aware models is streaming or online recommendation (He et al. 2016; Chang et al. 2017; Wang et al. 2018b), where we can access only the most recent data but not the complete historical data. Training only on recent data inevitably shifts the model to be more biased toward such data than the past. However, this poses a problem in retaining past information (Wang et al. 2018a).

Meta-learning in Recommender Systems

With the concept of meta-learning that aims to make learning faster, Vartak et al. (2017); Lee et al. (2019); Wei et al. (2020); Wang, Ding, and Caverlee (2021) adopted meta-learning techniques to alleviate the *cold-start* problem, where a model has not yet gathered sufficient information to draw any inferences for new users or new items. Liu et al. (2020) applied meta-learning for session-based recommendation (Hidasi et al. 2016). Luo et al. (2020) considered an application of meta-learning to select a suitable model for each user. Most of these approaches are based on MAML, which has been shown to be effective for few-shot learning.

Meta-learning also has been applied to improve recommendation in online learning scenario. For example, Zhang et al. (2020) proposed SML, a CNN-based *meta-update* method that uses MAML for meta-training; Xie et al. (2021) modified MAML for enhancing time adaptation of their recommender system.

[1]Gradient trajectory is the set of directions in the gradient path.

3 Preliminaries

Problem setup Let $\mathcal{U} = \{u_1, u_2, \dots, u_U\}$ be a finite set of users, $\mathcal{I} = \{i_1, i_2, \dots, i_I\}$ be a finite set of items, and $\mathcal{T} = (t_1, t_2, \dots, t_T)$ be a finite sequence of timestamps. The cardinalities of these sets correspond to the number of users $U = |\mathcal{U}|$, number of items $I = |\mathcal{I}|$, and number of timestamps $T = |\mathcal{T}|$. Given sets of interactions between users and items in each timestamps: $\mathcal{D} = \{\mathcal{D}_{t_1}, \mathcal{D}_{t_2}, \dots, \mathcal{D}_{t_T}\}$, where $t \in \mathcal{T}$ and $\mathcal{D}_t \in 2^{\mathcal{U}\times\mathcal{I}}$. The goal is to build a learning model to predict new interactions after t_T with high accuracy. In practice, such a model is evaluated using a test dataset $\{\mathcal{D}_{t_{T+1}}, \mathcal{D}_{t_{T+2}}, \dots\}$.

Objective function for recommendation To model how likely a user $u \in \mathcal{U}$ would interact with an item $i \in \mathcal{I}$, we define $f_\theta(u,i)$ to be a scoring function parameterized by θ. Commonly, Bayesian personalized ranking (BPR) (Rendle et al. 2009) is a suitable choice for the optimization target. The core idea is to maximize the interaction likelihood by using the contrastive learning principle. The model f_θ is trained to differentiate between the rankings of observed (positive) and unobserved (negative) interactions. Formally, we learn the recommendation model f_θ by minimizing the following BPR loss with respect to parameter θ.

$$\mathcal{L}(\theta; \mathcal{D}) = \mathop{\mathbb{E}}_{(u,i)\sim\mathcal{D},(u,j)\sim\tilde{\mathcal{D}}} [-\sigma\left(f_\theta(u,i) - f_\theta(u,j)\right)], \quad (1)$$

where $\sim$ denotes i.i.d. uniform samplings, $\tilde{\mathcal{D}}$ is the negative of $\mathcal{D}$: $\tilde{\mathcal{D}} = (\mathcal{U} \times \mathcal{I}) \setminus \mathcal{D}$, and σ is the sigmoid function. Throughout this paper, we use BPR as the loss function.

Leap: Trajectory-based Meta-Learning

Gradient-based meta-learning aims to obtain an initialization of parameters, also named meta-parameters $\bar{\theta}$, such that it allows fast adaptation across different learning tasks via the gradients (Finn, Abbeel, and Levine 2017). Flennerhag et al. (2019b) recently proposed a trajectory-based meta-learning method that minimizes the length of the learning process. The length of the learning process consists of (i) the distance between the loss landscape and (ii) the distance in the parameter space of an update, namely from θ_τ^k to θ_τ^{k+1} with $\theta_\tau^0 \leftarrow \bar{\theta}$. That results in the meta-gradient

$$\nabla F_{\bar{\theta}} = \mathbb{E}_{\tau\sim p(\tau)} \left[\sum_{k=0}^{K-1} (\underbrace{\Delta\mathcal{L}_\tau^k \nabla\mathcal{L}_\tau(\theta_\tau^k)}_{\text{(i)}} + \underbrace{\Delta\theta_\tau^k}_{\text{(ii)}}) \right], \quad (2)$$

where τ represents a task sampled from some distribution $p(\tau)$, K is the number of update steps, $\mathcal{L}_\tau$ is the loss with respect to τ, $\Delta\mathcal{L}_\tau^k = \mathcal{L}_\tau(\theta_\tau^{k+1}) - \mathcal{L}_\tau(\theta_\tau^k)$, and $\Delta\theta_\tau^k = \theta_\tau^{k+1} - \theta_\tau^k$. Note that the Jacobian term is omitted from our equation because using the identity matrix should provide a good approximation (Flennerhag et al. 2019b). Subsequently, we define the update of θ by adopting a simple gradient-based update, following MAML: $\theta_\tau^{k+1} \leftarrow \theta_\tau^k - \alpha\nabla\mathcal{L}_\tau(\theta_\tau^k)$, where α denotes the learning rate.

4 Proposed Method

4.1 Leaping Through Time

We adopt Leap to our recommendation models by considering data from each time step as a task. This section proposes two distinct components of LeapRec named Global Time Leap (GTL) and Ordered Time Leap (OTL) We denote $\bar{\gamma}$ and γ to be meta-parameters and model parameters of GTL while denoting $\bar{\omega}$ and ω to be meta-parameters and model parameters of OTL.

Algorithm 1: Meta-optimization for $\bar{\gamma}$ and $\bar{\omega}$ using Leap

```
Require: α, β, η: learning rate parameters
initialize γ̄ and ω̄
while not done do
    ∇F_γ̄ ← 0, ∇F_ω̄ ← 0, ω' ← ω̄
    for all t ∈ T do
        γ_t^0 ← γ̄, ω_t^0 ← ω'
        for k ← 0 to K − 1 do
            γ_t^{k+1} ← γ_t^k − α∇_{γ_k^t} L_t({γ_t^k, ω_t^k})
            ω_t^{k+1} ← ω_t^k − α∇_{ω_t^k} L_t({γ_t^k, ω_t^k})
        end for
        ω' ← ω_t^K
    end for
    update ∇F_γ̄, ∇F_ω̄ using Eq. (2).
    γ̄ ← γ̄ − (β/T)∇F_γ̄, ω̄ ← ω̄ − η∇F_ω̄
end while
```

Global Time Leap (GTL) GTL aims to find common patterns across all time steps by minimizing their gradient paths starting from $\bar{\gamma}$ (see Figure 1 for an illustration). We define a *task* as the recommendation objective under each timestamp with the following loss function: $\mathcal{L}_t(\gamma) = \mathcal{L}(\gamma; \mathcal{D}_t)$. The gradient update can be defined as: $\gamma_t^{k+1} \leftarrow \gamma_t^k - \alpha\nabla\mathcal{L}_t(\gamma_t^k)$ with $\gamma_t^0 \leftarrow \bar{\gamma}$. Because the parameters of GTL will revert to $\bar{\gamma}$ that stores the information across all timestamps before each update, this mechanism imitates the *static* or long-term preferences of users across all time steps.

Ordered Time Leap (OTL) While it is straightforward to use meta-learning in this context by assuming each time step is independent, encoding the sequential nature of temporal data is a non-trivial problem. Our idea is to capture the temporal information using the gradient trajectory that accumulates across the time sequence. We denote $\omega_{t_i}^k$ to be the value of ω at gradient step k and time step t_i. The meta-parameters $\bar{\omega}$ serves as the starting point for $\omega_{t_1}^0$. We update the model parameters and accumulate the gradient along the trajectory as follows. Within the same time step t_i, we update with learning rate α: $\omega_{t_i}^K \leftarrow \omega_{t_i}^0 - \alpha\sum_{k=0}^{K-1}\nabla\mathcal{L}_{t_i}(\omega_{t_i}^k)$; then, carry the result to the next time step t_{i+1}: $\omega_{t_{i+1}}^0 \leftarrow \omega_{t_i}^K$. This procedure is repeated until the last time step t_T.

The intuition behind this construction is that the obtained parameter $\omega_{t_T}^K$ is "close" to previous steps along the time sequence. Since the effect of catastrophic forgetting in neural networks can be mitigated with gradient trajectory regularization (Goodfellow et al. 2015; Chaudhry et al. 2018), the

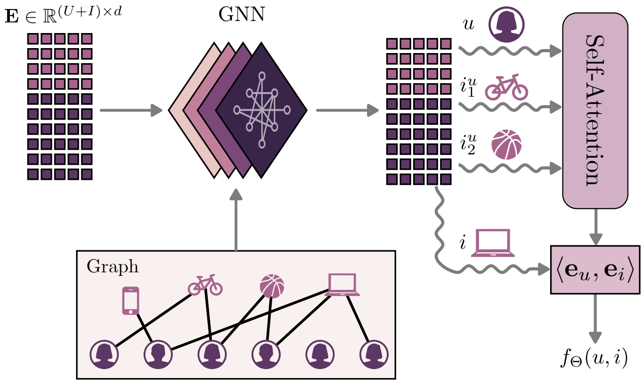


Figure 2: Our neural recommender. Θ denotes a set of embedding $\mathbf{E}$, GNN, and Self-Attention parameters.

shortest gradient path helps to retain useful information from previous time steps.

Meta-optimization LeapRec combines GTL and OTL by taking the summation of their prediction scores:

$$f_{\{\gamma,\omega\}}(u,i) = f_\gamma^{\mathsf{GTL}}(u,i) + f_\omega^{\mathsf{OTL}}(u,i). \tag{3}$$

To train $\{\bar{\gamma}, \bar{\omega}\}$, we simultaneously update $\{\gamma, \omega\}$ and update $\{\bar{\gamma}, \bar{\omega}\}$ at the end of an iteration using Leap, as shown in line 13 of Algorithm 1. In deployment, we use the latest model parameters $\{\gamma_{t_T}^K, \omega_{t_T}^K\}$ for making recommendations.

4.2 Neural Recommender

To demonstrate the application of LeapRec to a recommender system, we define a neural recommender described in Figure 2. The architecture consists of two main components: a graph neural network (GNN) (Scarselli et al. 2009; Kipf and Welling 2016) and a self-attention mechanism (SA) (Vaswani et al. 2017).

The neural recommender learns an embedding matrix $\mathbf{E} \in \mathbb{R}^{(U+I)\times d}$, where each row corresponds to a d-dimensional embedding of a user ($\mathbf{e}_u \in \mathbb{R}^d$) or an item ($\mathbf{e}_i \in \mathbb{R}^d$). Note that GTL and OTL can have different numbers of dimensionalities denoted by d^{GTL} and d^{OTL}, respectively. The GNN component refines initial features of users and items based on their related information guided by an interaction graph. On the other hand, the SA component extracts information from the histories of users to enhance user embeddings for more accurate recommendations.

Graph neural network Let $G = (\mathcal{V}, \mathcal{E})$ be a graph of user-item interactions. The node set $\mathcal{V}$ consists of all users and items in data $\mathcal{D}$, and the edge set $\mathcal{E}$ consists of interactions between users and items. Then, we use the GNN to propagate the node information based on the graph structure. The propagation for any node, namely m, in the graph by the ℓ^{th} layer of the GNN can be written as:

$$\mathbf{e}_m^\ell \leftarrow \sigma(a_{m,m}^\ell \mathbf{W}_1^\ell \mathbf{e}_m^{\ell-1} + \sum_{n\in\mathcal{N}_m} a_{m,n}^\ell \mathbf{W}_2^\ell \mathbf{e}_n^{\ell-1}), \tag{4}$$

where $\mathcal{N}_m$ denotes a set of the neighborhood of node m, σ denotes a non-linearity function, $\mathbf{W}_1^\ell, \mathbf{W}_2^\ell$ are transformation matrices of the ℓ^{th} layer, and $a_{m,n}^\ell$ is a normalizing factor; for example, GCN (Kipf and Welling 2016) uses $a_{m,n}^* = 1/\sqrt{|\mathcal{N}_m||\mathcal{N}_n|}$; or GAT (Veličković et al. 2018) proposed to use an attention mechanism to train $a_{m,n}^*$. The final outputs are stored in the embedding matrix $\mathbf{E}$.

Self-Attention We construct the sequence of items that a user, namely u, has been interacted with before timestamps t: $\mathcal{S}^u = (i_1^u, i_2^u, \ldots, i_{|\mathcal{S}^u|}^u)$. Embedding lookup is performed on E for each item in the sequence: $\mathbf{x}_j^u = \mathbf{e}_{i_j^u}$. We also append the embedding of user u to the sequence: $\mathbf{x}_{|\mathcal{S}^u|+1}^u = \mathbf{e}_u$. Then, we use SA to fuse the information inside the sequence. The attention score between inputs, namely from $\mathbf{x}_k^u$ to $\mathbf{x}_j^u$, and the output is calculated as

$$a_{k\to j}^u = \frac{(\mathbf{W}^Q\mathbf{x}_j^u)^\intercal(\mathbf{W}^K\mathbf{x}_k^u)}{\sqrt{d}}, \; \tilde{a}_{k\to j}^u = \frac{a_{k\to j}^u}{\sum_{k'=1}^{|\mathcal{S}^u|+1} a_{k'\to j}^u},$$
$$\mathbf{z}_j^u = \sum_{k=1}^{|\mathcal{S}^u|+1} \tilde{a}_{k\to j}^u \mathbf{W}^V \mathbf{x}_k^u,$$

where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d\times d}$ are the transformation matrices. Note that our SA cannot be aware of the positions in a sequence[2], unlike those SA-based methods for the sequential recommendation. After each SA layer, we apply each feature vector to a two-layer fully-connected feed-forward network (FFN) with ReLU activation. For each SA and FFN, we use a residual connection, dropout, and layer normalization, following Vaswani et al. (2017):

$$\mathbf{x}_j^u \leftarrow \text{LayerNorm}(\text{Dropout}(\mathbf{z}_j^u) + \mathbf{x}_j^u).$$

Prediction Thus far, we have defined the components of the neural recommender (Figure 2). To find the compatibility score between a user u and an item i, we obtain the embedding $\mathbf{e}_u$ from the last element of the sequence $\mathbf{x}_{|\mathcal{S}^u|+1}^u$, and embedding $\mathbf{e}_i$ from the output of the GNN. The score between u and i is given by the dot product $\langle \mathbf{e}_u, \mathbf{e}_i \rangle_{\mathbb{R}^d}$. This is a concrete modeling for f_ω^{OTL} and f_γ^{GTL} in Eq. 3.

Our neural recommender with LeapRec is a powerful model that unifies multiple concepts of recommender systems; GNN signifies the similarity between users and items beyond local interaction structures, which helps in learning common patterns; SA captures dependencies between items; LeapRec enhances both aforementioned to learn temporal dynamics. This is achievable because of the minimal computation cost of LeapRec.

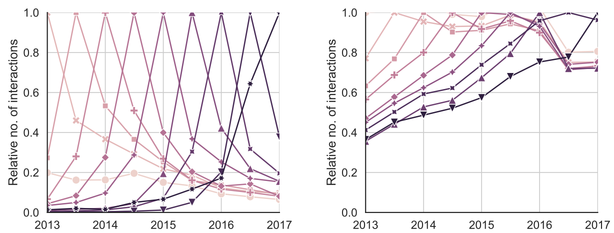
5 Experiments

We conducted experiments to answer the following research questions: **RQ1** How well does LeapRec perform compared with SOTA models? **RQ2** How effective is LeapRec as a plug-ins method for GNN and SA models? **RQ3** How important are GTL and OTL, and what are the changes in their learned representations over time? **RQ4** How sensitive is LeapRec to hyperparameters, and can LeapRec converge?

[2]The purpose is to highlight the contribution of LeapRec.

Dataset	# Users	# Items	# Actions	Cutting time
Amazon	70,097	56,708	1,540,911	2017/06
Goodreads	16,862	20,633	1,762,094	2015/12
Yelp	234,664	45,032	3,880,015	2018/12

Table 1: Basic statistics of datasets. Cutting times determine train/test splits.



(a) Amazon (b) Yelp

Figure 3: Relative number of interactions of groups of popular items (top 100) under each timestamp. Each line represents a group of items that became popular at the same time. The popularity of items on Amazon changed more drastically than one on Yelp.

Experimental Settings

Dataset We used three publicly available datasets: Amazon (Ni, Li, and McAuley 2019), Goodreads (Wang and Caverlee 2019), and Yelp[3]. User-item interactions in the Amazon dataset are product reviews. Our experiments used the preprocessed Amazon data by Wang et al. (2020), where data across categories are mixed. The Goodreads dataset is mined from the book-reading social network Goodreads. User-item interactions here are ratings and reviews of books. Yelp is a crowd-sourced business reviewing website. We used the latest officially provided data, updated on February 2021. For each dataset, we set a cutting time that only the data occurred before the time was used for training and the rest for validation/testing. The basic data statictics are presented in Table 1. To visualize the difference in the temporal dynamics between datasets, we plot the relative number of interactions in each group representing the top 100 items grouped by *peak* time on Amazon and Yelp. Figure 3 shows that popular items on Amazon changed more frequently compared with those on Yelp.

Evaluation protocol We divided each dataset into training, validation, and test sets based on the timestamps of interactions. We kept the interactions within six months after the cutting time as the validation set, and the rest after that as the test set. For each interaction in the test set, we randomly sampled 99 negatives instead of ranking all items to reduce the computation cost in the evaluation, and subsequently, the positive will be ranked among them by a model. We measured the recommendation quality using commonly used metrics for Top-K recommendation including Hit Rate (HR@K), Normalized Discounted Cumulative Gain (NDCG@K), and Mean Reciprocal Rank (MRR).

[3]https://www.yelp.com/dataset, updated February 2021.

Performance Comparison (RQ1)

To answer **RQ1**, we extensively evaluated LeapRec by comparing it to various types of recommendation models:

- ■ **Static models** are unaware of the temporal change in the behaviors of users. This category includes matrix factorization (MF) (Koren, Bell, and Volinsky 2009), deep neural network-based: NCF (He et al. 2017), variational autoencoder-based: BiVAE (Truong, Salah, and Lauw 2021), and GNN-based: NGCF (Wang et al. 2019b) and LightGCN (He et al. 2020) methods.
- ⏩ **Sequential models** are partially aware of the dynamic in the user behaviors based on the change in input sequences, this includes RNN-based approaches: GRU4Rec (Hidasi et al. 2016) and NARM (Li et al. 2017), a Memory-Network-based approach: SUM (Lian et al. 2021), a SA-based model: SASRec (Kang and McAuley 2018), and a meta-learning-based method: MetaTL (Wang, Ding, and Caverlee 2021).
- 🕓 **Temporal models** are explicitly aware of temporal dynamics or include temporal data in learning representations. This includes GNN-based: TGN (Rossi et al. 2020), RNN-based: SLi-Rec (Yu et al. 2019b), SA-based: TiSASRec (Li, Wang, and McAuley 2020) and GNN with SA-based: HyperRec (Wang et al. 2020)
- ▶ **Streaming models** inevitably favor recent data owing to the limited data access. We presents the results for SPMF (Wang et al. 2018b) and SML (Zhang et al. 2020). Note that SML uses FOMAML to train a meta-updater.

Hyper-parameter settings For all baselines, we conducted hyperparameter search on the dimensionality of embedding: $\{64, 128\}$, learning rate: $\{0.001, 0.0001\}$, and dropout: $\{0, 0.2, 0.5\}$. We also run hyperparameter search on the number of layers L: $\{1, 2, 3, 4\}$ for GNN and $\{1, 2\}$ for SA. We used the default values as provided for the rest. We used Adam (Kingma and Ba 2014) as the optimizer for all models. For LeapRec, we used the same hyperparameter settings as SASRec for its SA, and for its GNN, we adopt two-layer GCN (Kipf and Welling 2016) with dropout rate set at 0.2. We performed grid search on meta learning rates, the number of update steps K, and the time granularity on Amazon, then used them on other datasets[4]. The grid search for dimensionalities of OTL and GTL was performed for each dataset with values $\{0, 40, 91, 121, 128\}$ (see Table 4).

Experimental results We report the experimental results in Table 2. **LeapRec consistently outperforms all the baselines across all datasets and metrics.** Specifically, LeapRec achieves 8% to 18% relative improvements over the best performing baseline on Amazon dataset. This indicates that LeapRec can adapt fast enough to capture the

[4]After the grid search, we set learning rates: $\beta = \eta = 0.01$. For the best performance to answer **RQ1**, we set K to 40 and time granularity to one month, whereas for faster training to answer **RQ2-4**, we set K to 20 and the time granularity to two months.

Model	Amazon				Goodreads				Yelp			
	HR@1	HR@5	NDCG@5	MRR	HR@1	HR@5	NDCG@5	MRR	HR@1	HR@5	NDCG@5	MRR
■ MF	0.0820	0.2413	0.1629	0.1756	0.2008	0.5215	0.3662	0.3525	0.2847	0.6637	0.4823	0.4544
■ NCF	0.0751	0.2239	0.1506	0.1638	0.2003	0.5158	0.3630	0.3498	0.2923	0.6620	0.4857	0.4558
■ BiVAE	0.0891	0.2587	0.1748	0.1868	0.1966	0.5021	0.3542	0.3429	0.3056	0.6672	0.4954	0.4678
■ NGCF	0.0751	0.2273	0.1523	0.1646	0.1978	0.5146	0.3612	0.3482	0.2794	0.6545	0.4751	0.4463
■ LightGCN	0.0883	0.2517	0.1713	0.1826	0.2114	0.5342	0.3785	0.3632	0.3254	0.6930	0.5186	0.4875
⏩ GRU4Rec	0.0851	0.2681	0.1777	0.1890	0.2087	0.5194	0.3694	0.3569	0.3159	0.7145	0.5252	0.4895
⏩ NARM	0.0890	0.2764	0.1838	0.1945	0.2171	0.5340	0.3811	0.3669	0.3124	0.7160	0.5239	0.4879
⏩ SUM	0.0938	0.2896	0.1928	0.2025	0.2070	0.5224	0.3697	0.3563	0.2858	0.6906	0.4963	0.4627
⏩ SASRec	0.0879	0.2714	0.1806	0.1917	0.2147	0.5254	0.3756	0.3627	0.3181	0.7215	0.5295	0.4932
⏩ MetaTL**	0.0748	0.2267	0.1515	0.1652	0.1930	0.4989	0.3506	0.3397	0.2849	0.6704	0.4860	0.4560
🕓 TGN	0.1025	0.3227	0.2140	0.2200	0.1623	0.4448	0.3077	0.3021	0.2486	0.6536	0.4576	0.4278
🕓 SLi-Rec	0.0980	0.3005	0.2007	0.2097	0.2264	0.5509	0.3947	0.3785	0.2999	0.7103	0.5140	0.4781
🕓 TiSASRec	0.0888	0.2755	0.1832	0.1942	0.2176	0.5337	0.3814	0.3674	0.3117	0.7106	0.5205	0.4850
🕓 HyperRec	0.1056	0.3023	0.2057	0.2143	0.2143	0.5282	0.3769	0.3633	0.2793	0.6968	0.4966	0.4608
▶ SPMF	0.0587	0.1867	0.1232	0.1364	0.1366	0.3402	0.2415	0.2430	0.1526	0.3851	0.2731	0.2681
▶ SML*	0.1029	0.3031	0.2045	0.2115	0.1892	0.4776	0.3377	0.3294	0.2211	0.5533	0.3923	0.3785
LeapRec	**0.1252**	**0.3484**	**0.2391**	**0.2443**	**0.2452**	**0.5760**	**0.4171**	**0.3991**	**0.3420**	**0.7466**	**0.5547**	**0.5165**
Improvement	18.59%	7.97%	11.74%	11.07%	8.30%	4.55%	5.67%	5.44%	5.09%	3.48%	4.76%	4.72%

Table 2: Performance comparison of different models. The best performances are highlighted in bold. The second best performances are underlined. *Improvement* indicates *relative* improvement of LeapRec over the best performing baseline in each case. The symbols ■, ⏩, 🕓, and ▶ denote static, sequential, temporal, and streaming models, respectively. We mark MAML approaches by ** for vanilla MAML and * for first-order MAML (FOMAML).

dynamics between timestamps. Similarly, we observe that temporal models tend to outperform static and sequential models on the Amazon dataset. Conversely, on Yelp dataset, they achieve lower performances than those of sequential models. This observation demonstrates the inflexibility of the current temporal-aware SOTA recommendation models in modeling temporal dynamics. However, LeapRec performs well in both all datasets, indicating its flexibility.

LeapRec as a Plug-ins Method (RQ2)

Recently, GNN- and SA-based models have gained growing attention from researchers. It is interesting to determine the effect of LeapRec on these neural networks as a plug-ins method. In this experiment, we show that LeapRec can be applied to various types of models, making them temporal-aware. In the GNN-based category, we applied LeapRec on GCN, NGCF, and LightGCN, and compare the recommendation results between these GNNs with and without LeapRec. The main difference between these GNNs is based on feature transformation. LightGCN, a SOTA GNN for recommendation, is a simple GNN without feature transformation and nonlinearity. We also investigate the importance of Leap by introducing a FOMAML variant based on the modification of Eq. 2.

We plot the results of the GNN experiments in Figure 4. The NDCG@5 scores significantly increase up to 46.27% and 53.57% in relative gain among GCN and NGCF thanks to LeapRec. For LightGCN, we observe an improvement on the Amazon dataset and a slight decrease on Goodreads. We suppose that the feature transformation can accelerate the task-learning process (Lee and Choi 2018; Flennerhag et al. 2019b). In addition, **the LeapRec variants significantly outperform the FOMAML variants in all cases.** This empirically confirms the importance of trajectory-based meta-learning in the recommendation context.

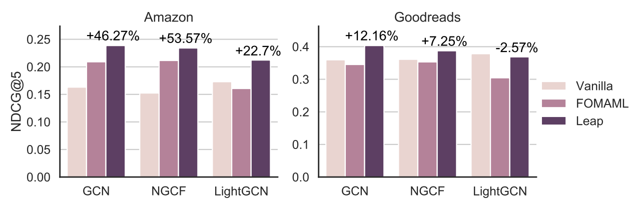


Figure 4: Performance of GNNs. FOMAML and Leap variants were equipped with GTL and OTL. The number over a bar indicates the relative improvement over its vanilla.

Model	Amazon		Goodreads	
	L=1	L=2	L=1	L=2
SASRec	0.1702	0.1806	0.3701	0.3709
TiSASRec	0.1774	0.1832	0.3709	0.3814
LeapRec-SASRec	0.1992	**0.2045**	**0.3956**	0.3934
Improvement	12.26%	11.63%	6.66%	3.15%

Table 3: Performance (NDCG@5) of SASRec without and with LeapRec with different numbers of layers L

The results of experiments on SA-based models are reported in Table 3. Even though SA models can already adapt to recent timestamps, LeapRec still improves SA with up to 12.26% gain over temporal-aware TiSASRec. This indicates the benefit of LeapRec's adaptability. We observe similar improvements to GNN and SA models on the Yelp dataset (omitted due to the page limit).

Dimensionality (d)		Dataset		
GTL (d^{GTL})	OTL (d^{OTL})	Amazon	Goodreads	Yelp
128	unused	0.2220	0.3885	0.5238
121	40	0.2339	0.4061	**0.5485**
91	91	0.2384	0.4025	0.5433
40	121	0.2415	**0.4093**	0.5366
unused	128	**0.2416**	0.3974	0.5379

Table 4: LeapRec with different combinations of GTL and OTL dimensionalities with similar FLOPs in total.

Figure 5: Shift in the item representations monitored on OTL and GTL of popular items under each timestamp. Each line represents a group of items that became popular at the same time (corresponding to the same group in Figure 3).

Analysis on Global and Ordered Time Leap (RQ3)

We investigate the effects of GTL and OTL on the performance of the model and the best resource-allocation ratio between them. We run experiments of LeapRec with varying the dimensionalities of GTL and OTL while maintaining the floating-point operations per second (FLOPs) of the model. The results are reported in Table 4. We observe that the best configuration depends on the datasets. In fast dynamic data such as Amazon, the model can perform well without GTL. Conversely, in slower dynamic data (Goodreads, Yelp), GTL becomes more important.

To further investigate the difference, we monitored a shift in the final representation of popular items in both GTL and OTL, as motivated by the observation in Figure 3. The shift of the representation of item i from time t_{j-1} to t_j is defined as: $s^i_{t_j} = \|\hat{\mathbf{e}}^K_{t_j,i} - \hat{\mathbf{e}}^K_{t_{j-1},i}\|^2$, where $\hat{\mathbf{e}}$ denotes the normalized vector of $\mathbf{e}$. Remind that K is the number of update steps.

We plot the shift monitored on OTL and GTL in Figure 5. For OTL, we observe that the popular group at a certain time had a larger shift than less popular groups. This implies that OTL *actively* adjusted such a group more than others to adapt to the trends shown in Figure 3. As time went on, the popularity of such a group would decrease or become stable, and OTL also became less active accordingly. In addition, we observe that OTL was more active on Amazon than Yelp because of the difference in the magnitude of temporal dynamics. On the other hand, for GTL, its activeness was almost constant across time. These demonstrate that GTL holds common and long-term patterns while OTL captures temporal information that varies frequently.

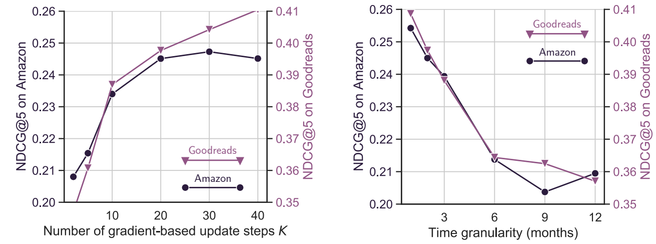


Figure 6: Hyperparameter sensitivity analyses

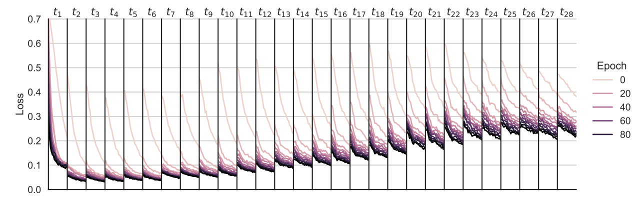


Figure 7: Training loss of LeapRec in each timestamp on Amazon dataset in different epochs. The horizontal axis in each time window is the gradient-based update steps.

Sensitivity and Convergence Analysis (RQ4)

We investigate the effect of the number of gradient-based update steps K and time granularity of timestamps on the performance of LeapRec. We conducted further experiments using LeapRec with varying numbers of both and plot the results in Figure 6. We observe that the performance of LeapRec tends to increase with the number of update steps. The performance on Goodreads continued to increase when $K > 20$, but reached a plateau on Amazon. This possibly can be contributed to the difference in the densities of the datasets. With respect to time granularity, LeapRec performs best with the finest-grained of one month. This implies its fast adaptability. Unsurprisingly, LeapRec behaves closer to a static model with coarser-grained time granularity.

Finally, we analyze the convergence of LeapRec by monitoring its training loss. We plot the training loss across timestamps in Figure 7. We observe that the training loss continued to decrease on all timestamps, suggesting that LeapRec could converge.

6 Conclusion

In this work, we proposed a novel method using trajectory-based meta-learning to solve the temporal dynamic challenge in modern recommender systems. The empirical results clearly show the advantages of our method compared with a wide range of SOTA methods. Our deep analyses into the temporal dynamics of benchmark datasets also validate our designs of GTL and OTL as they capture long-term and short-term information respectively. We believe the effectiveness of our proposal will initiate new discussions regarding trajectory-based recommendation.

Acknowledgments

We would like to thank Noppayut Sriwatanasakdi and Nontawat Charoenphakdee for helpful discussion. This work was partially supported by JST CREST (Grant Number JPMJCR1687), JSPS Grant-in-Aid for Scientific Research (Grant Number 17H01785, 21K12042), and the New Energy and Industrial Technology Development Organization (Grant Number JPNP20006). Nuttapong Chairatanakul was supported by MEXT scholarship.

References

Aggarwal, C. C. 2016. Model-Based Collaborative Filtering. In *Recommender Systems: The Textbook*, 71–138. Springer International Publishing.

Chairatanakul, N.; Murata, T.; and Liu, X. 2019. Recurrent Translation-Based Network for Top-N Sparse Sequential Recommendation. *IEEE Access*, 7: 131567–131576.

Chang, S.; Zhang, Y.; Tang, J.; Yin, D.; Chang, Y.; Hasegawa-Johnson, M. A.; and Huang, T. S. 2017. Streaming Recommender Systems. In *WWW '17*, 381–389.

Chaudhry, A.; Dokania, P. K.; Ajanthan, T.; and Torr, P. H. S. 2018. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In Ferrari, V.; Hebert, M.; Sminchisescu, C.; and Weiss, Y., eds., *ECCV 2018*, volume 11215, 556–572.

Chen, X.; Xu, H.; Zhang, Y.; Tang, J.; Cao, Y.; Qin, Z.; and Zha, H. 2018. Sequential Recommendation with User Memory Networks. In *WSDM '18*, 108–116.

Fan, Z.; Liu, Z.; Zhang, J.; Xiong, Y.; Zheng, L.; and Yu, P. S. 2021. Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer. In *CIKM '21*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML '17*, 1126–1135.

Flennerhag, S.; Moreno, P. G.; Lawrence, N. D.; and Damianou, A. 2019a. Transferring Knowledge across Learning Processes. In *ICLR*.

Flennerhag, S.; Rusu, A. A.; Pascanu, R.; Visin, F.; Yin, H.; and Hadsell, R. 2019b. Meta-Learning with Warped Gradient Descent. In *ICLR*.

Goodfellow, I. J.; Mirza, M.; Xiao, D.; Courville, A.; and Bengio, Y. 2015. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv:1312.6211 [cs, stat]*.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR '20*, 639–648.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural Collaborative Filtering. In *WWW '17*, 173–182.

He, X.; Zhang, H.; Kan, M.-Y.; and Chua, T.-S. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *SIGIR '16*, 549–558.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-Based Recommendations with Recurrent Neural Networks. In *ICLR*.

Ji, W.; Wang, K.; Wang, X.; Chen, T.; and Cristea, A. 2020. Sequential Recommender via Time-Aware Attentive Memory Network. In *CIKM '20*, 565–574.

Kang, W.-C.; and McAuley, J. 2018. Self-Attentive Sequential Recommendation. In *ICDM '18*, 197–206.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Koren, Y. 2009. Collaborative Filtering with Temporal Dynamics. In *KDD '09*, 447–456.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8): 30–37.

Lee, H.; Im, J.; Jang, S.; Cho, H.; and Chung, S. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *KDD '19*, 1073–1082.

Lee, Y.; and Choi, S. 2018. Gradient-Based Meta-Learning with Learned Layerwise Metric and Subspace. In *ICML '18*, 2927–2936.

Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; and Ma, J. 2017. Neural Attentive Session-Based Recommendation. In *CIKM '17*, 1419–1428.

Li, J.; Wang, Y.; and McAuley, J. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM '20*, 322–330.

Lian, J.; Batal, I.; Liu, Z.; Soni, A.; Kang, E. Y.; Wang, Y.; and Xie, X. 2021. Multi-Interest-Aware User Modeling for Large-Scale Sequential Recommendations. *arXiv:2102.09211 [cs]*.

Liu, Q.; Zeng, Y.; Mokhosi, R.; and Zhang, H. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-Based Recommendation. In *KDD '18*, 1831–1839.

Liu, Z.; Chen, H.; Sun, F.; Xie, X.; Gao, J.; Ding, B.; and Shen, Y. 2020. Intent Preference Decoupling for User Representation on Online Recommender System. In *IJCAI-20*, 2575–2582.

Luo, M.; Chen, F.; Cheng, P.; Dong, Z.; He, X.; Feng, J.; and Li, Z. 2020. MetaSelector: Meta-Learning for Recommendation with User-Level Adaptive Model Selection. In *WWW '20*, 2507–2513.

Ma, C.; Ma, L.; Zhang, Y.; Sun, J.; Liu, X.; and Coates, M. 2020. Memory Augmented Graph Neural Networks for Sequential Recommendation. In *AAAI-20*, volume 34, 5045–5052.

Ni, J.; Li, J.; and McAuley, J. 2019. Justifying Recommendations Using Distantly-Labeled Reviews and Fine-Grained Aspects. In *EMNLP-IJCNLP 2019*, 188–197.

Nichol, A.; Achiam, J.; and Schulman, J. 2018. On First-Order Meta-Learning Algorithms. arXiv:1803.02999.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the Difficulty of Training Recurrent Neural Networks. In *ICML '13*, III–1310–III–1318.

Ravi, S.; and Larochelle, H. 2016. Optimization as a Model for Few-Shot Learning. In *ICLR*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI '09*, 452–461.

Ricci, F.; Rokach, L.; and Shapira, B. 2011. Introduction to Recommender Systems Handbook. In Ricci, F.; Rokach, L.; Shapira, B.; and Kantor, P. B., eds., *Recommender Systems Handbook*, 1–35. Springer US.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*.

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T. 2016. Meta-Learning with Memory-Augmented Neural Networks. In *ICML '16*, 1842–1850.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1): 61–80.

Schafer, J. B.; Konstan, J.; and Riedl, J. 1999. Recommender Systems in E-Commerce. In *EC '99*, 158–166.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM '19*, 1441–1450.

Truong, Q.-T.; Salah, A.; and Lauw, H. W. 2021. Bilateral Variational Autoencoder for Collaborative Filtering. In *WSDM '21*, 292–300.

Vartak, M.; Thiagarajan, A.; Miranda, C.; Bratman, J.; and Larochelle, H. 2017. A Meta-Learning Perspective on Cold-Start Recommendations for Items. In *NIPS'17*, 6907–6917.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention Is All You Need. In *NIPS '17*, volume 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Vilalta, R.; and Drissi, Y. 2002. A Perspective View and Survey of Meta-Learning. *Artificial Intelligence Review*, 18(2): 77–95.

Wang, J.; and Caverlee, J. 2019. Recurrent Recommendation with Local Coherence. In *WSDM '19*, 564–572.

Wang, J.; Ding, K.; and Caverlee, J. 2021. Sequential Recommendation for Cold-Start Users with Meta Transitional Learning. In *SIGIR '21*, 1783–1787.

Wang, J.; Ding, K.; Hong, L.; Liu, H.; and Caverlee, J. 2020. Next-Item Recommendation with Sequential Hypergraphs. In *SIGIR '20*, 1101–1110.

Wang, Q.; Yin, H.; Hu, Z.; Lian, D.; Wang, H.; and Huang, Z. 2018a. Neural Memory Streaming Recommender Networks with Adversarial Training. In *KDD '18*, 2467–2475.

Wang, S.; Hu, L.; Wang, Y.; Cao, L.; Sheng, Q. Z.; and Orgun, M. 2019a. Sequential Recommender Systems: Challenges, Progress and Prospects. In *IJCAI-19*, 6332–6338.

Wang, W.; Yin, H.; Huang, Z.; Wang, Q.; Du, X.; and Nguyen, Q. V. H. 2018b. Streaming Ranking Based Recommender Systems. In *SIGIR '18*, 525–534.

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019b. Neural Graph Collaborative Filtering. In *SIGIR '19*, 165–174.

Wei, T.; Wu, Z.; Li, R.; Hu, Z.; Feng, F.; He, X.; Sun, Y.; and Wang, W. 2020. Fast Adaptation for Cold-Start Collaborative Filtering with Meta-Learning. In *ICDM '20*, 661–670.

Weston, J.; Chopra, S.; and Bordes, A. 2015. Memory Networks. In *ICLR*.

Wu, C.-Y.; Ahmed, A.; Beutel, A.; Smola, A. J.; and Jing, H. 2017. Recurrent Recommender Networks. In *WSDM '17*, 495–503.

Xie, R.; Wang, Y.; Wang, R.; Lu, Y.; Zou, Y.; Xia, F.; and Lin, L. 2021. Long Short-Term Temporal Meta-Learning in Online Recommendation. *arXiv:2105.03686 [cs]*.

Xiong, L.; Chen, X.; Huang, T.-K.; Schneider, J.; and Carbonell, J. G. 2010. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *SIAM '10*, 211–222.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2019. Inductive Representation Learning on Temporal Graphs. In *ICLR*.

Ye, W.; Wang, S.; Chen, X.; Wang, X.; Qin, Z.; and Yin, D. 2020. Time Matters: Sequential Recommendation with Complex Temporal Information. In *SIGIR '20*, 1459–1468.

Yu, Y.; Si, X.; Hu, C.; and Zhang, J. 2019a. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, 31(7): 1235–1270.

Yu, Z.; Lian, J.; Mahmoody, A.; Liu, G.; and Xie, X. 2019b. Adaptive User Modeling with Long and Short-Term Preferences for Personalized Recommendation. In *IJCAI-19*, 4213–4219.

Zhang, Y.; Feng, F.; Wang, C.; He, X.; Wang, M.; Li, Y.; and Zhang, Y. 2020. How to Retrain Recommender System? A Sequential Meta-Learning Method. In *SIGIR '20*, 1479–1488.

Zhu, Y.; Li, H.; Liao, Y.; Wang, B.; Guan, Z.; Liu, H.; and Cai, D. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM. In *IJCAI-17*, 3602–3608.