2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00087

Incremental Learning for Multi-Interest Sequential Recommendation

Zhikai Wang
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, China
Cloudcatcher.888@sjtu.edu.cn

Yanyan Shen*
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, China
shenyy@sjtu.edu.cn

***Abstract*—In recent years, sequential recommendation has been widely researched, which aims to predict the next item of interest based on user's previously interacted item sequence. Existing works utilize capsule network and self-attention method to explicitly capture multiple underlying interests from a user's interaction sequence, achieving the state-of-the-art sequential recommendation performance. In practice, the lengths of user interaction sequences are ever-increasing and users might develop new interests from new interactions, and a model should be updated or even expanded continuously to capture the new user interests. We refer to this problem as incremental multi-interest sequential recommendation, which has not yet been well investigated in the existing literature. In this paper, we propose an effective incremental learning framework for multi-interest sequential recommendation called IMSR, which augments the traditional fine-tuning strategy with the existing-interests retainer (EIR), new-interests detector (NID), and projection-based interests trimmer (PIT) to adaptively expand the model to accommodate user's new interests and prevent it from forgetting user's existing interests. Extensive experiments on real-world datasets verify the effectiveness of the proposed IMSR on incremental multi-interest sequential recommendation, compared with various baseline approaches.**

***Index Terms*—Incremental learning, multi-interest sequential recommendation, capsule network, self-attention**

I. Introduction

Sequential recommendation, which aims at predicting user preference on items given historical user interaction sequence, is an essential task in nowadays recommender systems. Various approaches have been proposed for sequential recommendation [1]–[4]. They typically extract one user preference vector from the interaction sequence. Recently, many works [5]–[8] argued that users usually have multiple latent interests beneath their interaction sequences, and hence they proposed to incorporate different kinds of multi-interest extractors into sequential recommendation models, which is referred to as multi-interest sequential recommendation (MSR). Specifically, existing works employ the dynamic routing or self-attention mechanism to explicitly compute the user's multiple interest vectors based on sequentially interacted items. The final user preference vector is derived by aggregating multiple interest vectors and used for next item prediction.

In real recommender systems, the lengths of user interaction sequences are ever-increasing, and users might develop new interests from new interactions. Both existing and new interests might reappear in future interactions. As stated in the experiments of the previous work [8], over eighty percent of interests will reappear for more than three times. However, we do not know which of the existing or new interests will reappear. Therefore, it is of great importance to update an MSR model continuously to capture users' new interests while retaining all existing interests.

A simple strategy for model updating is to retrain the model using the whole user interaction sequences per time span (e.g., one hour or a day). In this strategy, both existing and newly developed interests in the user interaction sequence can be captured. However, training on the whole user interaction sequences is extremely time-consuming, and this strategy is not practical in situations where all the historical interactions are not obtainable or the memory space for maintaining the whole sequences during training and inference is constrained. To address this problem, a more cost-effective strategy is to fine-tune the model with the new interactions in the current time span and inherit the model parameters from the previous time span as the initial values. In this strategy, users' existing interests extracted from previous interactions are not thrown away completely and will be updated in the current time span.

By rethinking the basic idea of fine-tuning an MSR model, we have two intuitions for learning users' evolving interests over time, including the existing interests and the newly developed ones, respectively. On one hand, user's existing interests extracted from previous time spans are not fixed and may drift gradually over time. For instance, a user who was interested in flip phones may later adapt his/her interest to smartphones. On the other hand, the number of newly developed interests may change dynamically in different time spans. For example, a user, who has been interested in computer games for a long time, might suddenly become interested in baby care products in a new time span due to the birth of a child. Based on the above two intuitions, our goal of this paper is to develop an incremental multi-interest sequential recommendation framework which preserves existing interests via modest drifting or refining and adaptively increases the number of new interest vectors to deal with newly developed interests. To establish the incremental MSR framework mentioned above, there are three challenges remained to be tackled. First, existing interests

*Yanyan Shen is the corresponding author.

might be forgotten when only newly collected interactions are used for model training, which will result in performance degeneration on items related to existing interests. In our experiment, we find the performance will drop fast by only fine-tuning the model. Second, new interests conveyed by new interactions in the current time span may overlay existing interests due to the fixed model complexity, i.e., a fixed number of interest vectors over time spans. Likewise, existing interests might in turn affect the learning of new interests. Hence, it is important to prevent the learning of existing and new interests from interfering with each other. Third, different users will develop different numbers of new interests in a new time span. It is required to decide the number of new interests adaptively for each user during each time span.

The literature for general incremental learning includes reservoir-based methods, regularization-based methods and model-expansion methods. Reservoir-based methods [9]–[11] focus on preserving past knowledge by selecting samples from the reservoir for model updating based on prioritizing recency or the extent of being forgotten. However, they need to store historical interactions which may not be obtainable in incremental MSR. Regularization-based methods [12], [13] propose to preserve the knowledge learned in the past time spans by enforcing regularization terms, which regularly restrains the model parameters rather than user latent representations. Moreover, these kinds of methods do not vary model complexity over time, and they cannot generate new interest vectors which require extra model parameters. Model-expansion methods [14]–[16] aim to expand the model capacity to cope with new knowledge during incremental learning. While model expansion can be applied to capture new interests and prevent interests from interfering with each other, existing works [16], [17] require knowing the expanded model capacity at each time span in advance. Since the number of new interests for each user is dynamically changing, it is a non-trivial task to perform model expansion for incremental multi-interest sequential recommendation. To these ends, existing incremental learning methods are of limited use in tackling the challenges in incremental MSR.

In this paper, we develop an end-to-end framework for Incremental Multi-interest Sequential Recommendation (IMSR), which involves the existing-interests retainer, new-interests detector, and projection-based interests trimmer. More specifically, the existing-interest retainer, which aims to solve the first challenge, uses the distillation loss [18] to make sure that the existing interest vectors are not far away from their original positions. The new-interests detector is proposed to solve the second challenge, which determines when to create new interests based on the change in distribution of item numbers being classified to different interests. The projection-based interests trimmer is proposed to solve the third challenge, which first allocates a relatively large number of new interests and then modifies the learned interest vectors' magnitudes and directions to remove redundant interests. In this way, our framework can preserve the existing interests while an appropriate number of new interests are developed from the new interactions.

The main contributions are summarized as follows:

- We propose a framework for incremental multi-interest sequential recommendation which can dynamically capture new interests from recent interactions for each user while retaining existing interests.
- We develop an existing-interests retainer based on knowledge distillation which can preserve existing interests from heavily drifting. We design a new-interests detector to determine when to create new interests and a projection-based interests trimmer to develop new interests adaptively.
- We implement our framework on two kinds of multi-interest sequential recommendation models, i.e., dynamic-routing-based and self-attention-based models, and conduct extensive experiments on four real datasets. The results demonstrate that our proposed framework can adaptively increase interest vectors to accommodate new interests, effectively address the existing interets forgetting problem, and achieve competing performance as full retraining method compared with the existing incremental learning methods.

II. Background

Consider a recommendation dataset consisting of interaction records between a set $\mathcal{U}$ of users and a set $\mathcal{I}$ of items. Let $\mathcal{A} = \{(u, i, s)\}$ denote all the interactions in the dataset, where $u \in \mathcal{U}$ is a user id, $i \in \mathcal{I}$ is an item id, and s is the timestamp when the interaction happens. In the sequential recommendation setting, each instance consists of two parts. The input part is u's historical interaction sequence which is a sequence of u's previously interacted items chronically, denoted as $\mathcal{S}_u = \{i_{1,u}, i_{2,u}, \cdots, i_{n,u}\}$. The output part is the target items $i_{a,u}$ that the user u lastly interacts with. Multi-interest sequential recommendation (MSR) aims to learn a function f_{user} which can map a user's interaction sequence $\mathcal{S}_u$ into a multi-interest user representation $\boldsymbol{H}_u$:

$$\boldsymbol{H}_u = f_{\text{user}}(\mathcal{S}_u) = (\boldsymbol{h}_{u,1}, \cdots, \boldsymbol{h}_{u,K}) \in \mathbb{R}^{d \times K}, \qquad (1)$$

where $\boldsymbol{h}_{u,k}$ denotes the k^{th} interest vector of u ($k \in [1, K]$), K is the number of user interests, and d is the dimension of each interest vector. The MSR task is to find the top-N candidate items with the highest scores using a scoring function:

$$f_{\text{score}}(\boldsymbol{H}_u, \boldsymbol{e}_i), \qquad (2)$$

where $\boldsymbol{e}_i \in \mathbb{R}^{d \times 1}$ denotes the embedding of item $i \in \mathcal{I}$.

In practice, user interactions are prolonged with new interactions collected over time spans. Thus an MSR model should be retrained in each time span to update user representations. Basically, full retraining strategy can be used for MSR, which uses all the historical interaction sequences to retrain the model in each time span. As full retraining is time- and space-consuming when the historical interaction sequences are extremely long, a more cost-effective way is to perform incremental learning with new interactions collected in each time span, which can be formally defined as the incremental multi-interest sequential recommendation. Table I provides the key notations used throughout the paper.

TABLE I
NOTATION TABLE.

Notation	Description
(u, i, t)	the user u interacted with item i at timestamp t
i_a	the target item during training
$\mathcal{U}, \mathcal{I}$	the sets of users and items
$\mathcal{S}_u$	the historical behavior sequence of user u
$\boldsymbol{h}_{u,k}$	the k^{th} interest vector for user u
$\mathbf{H}_u$	the multi-interest representation of user u
$\boldsymbol{e}_a$	the target item's embedding

Definition 1 (Incremental multi-interest sequential recommendation): Let $\mathcal{S}_u^t = \{i_{1,u}^t, i_{2,u}^t, \cdots, i_{n,u}^t\}$ and $i_{a,u}^t$ denote the user u's new interactions and the target item within time span t, respectively. In time span t, we have an MSR model $\mathcal{M}^{t-1}$ trained at the previous time span and try to update the model to be $\mathcal{M}^t$ with newly collected sequences $\{\mathcal{S}_u^t | u \in \mathcal{U}\}$.

III. The Basic Fine-tuning Approach

To solve the above defined incremental MSR, we first describe the basic fine-tuning approach based on two kinds of MSR models, namely dynamic-routing based model (DR) and self-attention-based model (SA).

1) Dynamic-routing based MSR (DR): DR utilizes the dynamic routing mechanism to extract multiple interest vectors from user interaction sequences. For each user u we use $E_u^t = \{\boldsymbol{e}_i \mid (u,i) \in \mathcal{S}_u^t\}$ to denote the set of item embeddings involved in the user interaction sequence $\mathcal{S}_u^t$, and use $\boldsymbol{e}_a^t$ to denote the target item embedding. It adopts the Behavior-to-Interest (B2I) dynamic routing [5], [6] to learn a group of interests from a user's interaction sequence. Formally, each item embedding $\boldsymbol{e}_i \in E_u^t$ is first transformed into the low-level capsules' plane using a shared affine transforming matrix $\boldsymbol{W}^t \in \mathbb{R}^{d \times d}$ as:

$$\hat{\boldsymbol{e}}_i = \boldsymbol{W}^t \boldsymbol{e}_i. \tag{3}$$

Then a B2I dynamic routing procedure will be applied to the transformed low-level capsules for L iterations. $\boldsymbol{h}_k^{(0),t}$ is initialized as zero. The $k \in [1, ..., K_u^t]$ high-level capsule in the $l \in [1, ..., L]$ iteration is computed as:

$$\boldsymbol{h}_k^{(l),t} = \phi(\sum_{i=1}^{|E_u^t|} c_{ik}^{(l),t} \hat{\boldsymbol{e}}_i), \tag{4}$$

where $c_{ik}^{(l),t}$ is the l^{th} layer's vote from item i to interest k in time span t, which is the softmax result of $\boldsymbol{e}_i \boldsymbol{h}_k^{(l-1),t}$ over other items in E_u^t. In later sections, the interest capsule only refers to the last layer's capsule if not indicated. ϕ denotes the squash function [19], which leaves the direction of the input vector unchanged but decreases its magnitude. Intuitively, Eq. (4) softly clusters the low-level capsules into K_u^t different high-level capsules, which can be regarded as K_u^t different interests extracted from the user interaction sequence. Note that the value of K_u^t can be different for different users.

After performing the multi-interest extractor, we obtain K_u^t vectors $\boldsymbol{H}_u^t = \{\boldsymbol{h}_1^t, \cdots, \boldsymbol{h}_{K_u^t}^t\}$ representing user's different interests. In each time span, the target item embedding $\boldsymbol{e}_a^t$ is the query to evaluate the importance of each user interest. We then derive a target-item-sensitive user representation $\boldsymbol{v}_u^t \in \mathbb{R}^{d \times 1}$ at time span t by aggregating user interests in an attentive manner. Formally, we have:

$$\boldsymbol{v}_u^t = \sum_{k=1}^{K_u^t} \beta_k \boldsymbol{h}_k^t, \tag{5}$$

where β_k is the softmax value of $\boldsymbol{e}_a^t \boldsymbol{h}_k^t$ over all interests. We perform dot-product between $\boldsymbol{v}_u^t$ and the target item embedding $\boldsymbol{e}_a^t$ to produce the preference score, i.e., $\boldsymbol{v}_u^{t\top} \boldsymbol{e}_a^t$.

The objective function for time span t is:

$$\mathcal{L}_{SS}^t = \sum_{\mathcal{D}^t} \log \frac{\exp(\boldsymbol{v}_u^{t\top} \boldsymbol{e}_a^t)}{\sum_{i \in \mathcal{I}'} \exp(\boldsymbol{v}_u^{t\top} \boldsymbol{e}_i)}, \tag{6}$$

where $\mathcal{I}' \subset \mathcal{I} \backslash \{i_a^t\}$ is a small sampled subset of items excluding the target item i_a^t, which serves as the negative samples set [20]. DR uses backpropagation on loss function in Eq. (6) to update $\boldsymbol{W}^t$. In time span $t+1$, $\boldsymbol{W}^{t+1}$ will be fine-tuned based on $\boldsymbol{W}^t$ as the initial value. In this way, the user's existing interests are not discarded completely and can be refined with the incremental interaction sequence.

2) Self-attention based MSR (SA): Given the embeddings of user interactions $E_u^t \in \mathbb{R}^{d \times n}$, SA uses the self-attention mechanism to obtain a vector of weights $\boldsymbol{a}_u^t \in \mathcal{R}^n$ as:

$$\boldsymbol{a}_u^t = softmax(\boldsymbol{w}_u^{t\top} \tanh(\boldsymbol{W}_1^t E_u^t))^\top, \tag{7}$$

where $\boldsymbol{w}_u^t$ and $\boldsymbol{W}_1^t$ are trainable parameters with size d_a and $d_a \times d$, respectively.

The vector $\boldsymbol{a}_u^t$ with size n represents the attention weight of user interactions. When we sum up the embeddings of user interactions according to the attention weights, we can obtain a vector representation $\boldsymbol{h}_u^t = E_u^t \boldsymbol{a}_u^t$ for the user.

This vector representation reflects a specific interest of the user u. To represent the overall interests of the user, we need multiple $\boldsymbol{h}_u$ from the user interactions that focus on different interests. Thus we extend the $\boldsymbol{w}_u^t$ into a d_a-by-K matrix as $\boldsymbol{W}_u^t$. Then the attention vector becomes an attention matrix $\boldsymbol{A}_u^t$ as:

$$\boldsymbol{A}_u^t = softmax(\boldsymbol{W}_u^{t\top} \tanh(\boldsymbol{W}_1^t E_u^t))^\top. \tag{8}$$

Note that we assign different $\boldsymbol{w}_u$ to different users for computing their exclusive interest vectors. The final matrix of user interests $\boldsymbol{H}_u^t$ can be computed by:

$$\boldsymbol{H}_u^t = E_u^t \boldsymbol{A}_u^t. \tag{9}$$

Then SA performs the same aggregator in DR to compute the objective function and uses backpropagation to update $\boldsymbol{W}_1^t$ and $\{\boldsymbol{W}_u^t \mid u \in \mathcal{U}\}$. In time span $t+1$, $\boldsymbol{W}_1^{t+1}$ and $\{\boldsymbol{W}_u^{t+1} \mid u \in \mathcal{U}\}$ will be fine-tuned based on $\boldsymbol{W}_1^t$ and $\{\boldsymbol{W}_u^t \mid u \in \mathcal{U}\}$ to preserve the existing interests.

By far, we introduce a vanilla way to train MSR models in an incremental manner. However, existing interests might be forgotten using the fine-tuning method, which will result in performance degeneration on items related to existing interests. Moreover, new interests may appear in new interactions. If the model capacity is fixed, the occurrence of new interests may interfere with existing interests. Hence, it is desirable to adaptively increase the number of interests to accommodate new interests. In what follows, we describe how the proposed IMSR resolves these challenges.

IV. The IMSR Framework

A. Overview

Our proposed IMSR is an incremental learning framework for multi-interest sequential recommendation that can dynamically capture new interests from new interactions while retaining existing interests. Figure 1 depicts the overall architecture of IMSR which consists of two components. The first component is the base model, a dynamic routing or self-attention-based multi-interest recommendation model [6]. The second component contains three modules called existing-interests retainer (EIR), new-interests detector (NID), and projection-based new-interests trimmer (PIT), respectively. EIR is designed to prevent the existing interests learned by the base model in previous time spans from heavily drifting. The new-interests detector creates new interests to prevent the learning of existing and new interests from interfering with each other. The projection-based interests trimmer is proposed to decide the number of new interests adaptively for each user during each time span.

The remaining parts of this section will be organized as follows. We first introduce the EIR for interest retention in Section IV-B. We describe the NID for new-interests detection in Section IV-C and PIT for the new-interests expansion in Section IV-D. In Section IV-E and Section IV-F, we present the training/inference procedure and the implementation details, respectively.

B. Existing-interests Retention

As stated in the previous works [7], [8], existing and new interests might reappear in the future interactions and we do not know which of the existing or new interests will reappear. Therefore, it is of great importance to preserve the all existing interests from the previous time span and prevent them from changing greatly or being taken place by new interests. At the same time, the items belonging to the existing interests can be absorbed into these interests to adjust their representations. The most straightforward idea is to equip a distance-based regularization term like euclidean distance to prevent the existing interest vectors from drifting far away from their original representations after performing multi-interest extractor in a new time span. However, this method is not flexible enough. More specifically, the distance between two vectors of completely different interests can be small, which means even a little change in the euclidean space may change the semantics of an interest vector. Thus in worse cases, the distance-based regularization term will prevent items from being recommended by the corresponding interests but cannot prevent the existing interest from changing completely.

Here we find that knowledge distillation, which does not restrain the representation vector itself but focuses on the restraint of the output logits, is a more flexible way to overcome forgetting problems. Hinton et al. [18] used knowledge distillation to transfer knowledge from an ensemble of models into a single model for efficient deployment, where knowledge distillation loss is used to preserve knowledge from the cumbersome model by encouraging the outputs of distilled model to approximate that of the cumbersome model. Similarly, the authors of LwF [21] performed knowledge distillation to learn knowledge from new tasks while keeping knowledge on existing tasks in incremental learning scenarios.

Instead of restraining the existing interest representation vector $\mathbf{h}_k^{(L),t}$ $(k \in [1, ...K_u^{t-1}])$, we propose to use a knowledge distillation loss to overcome the problem. Vanilla distillation loss [18] needs a learned teacher model, which is not obtainable in MSR, to guide the student model. However, if we regard the interest capsules as several item classes, the target item preference scoring can also be regarded as a matching-based classification model. Similar to the original knowledge distillation idea, the existing interest vectors can be viewed as the parameters of the teacher model, and the new interest vectors are the parameters of the student model. Thus, a distillation loss to preserve the existing knowledge needs to encourage the outputs of the student model to approximate those of the teacher model. Formally, we have:

$$\mathcal{L}_{KD,k,u}^t = \mathcal{L}_{CE}\left(\sigma\left(\frac{f(\boldsymbol{h}_k^t, \boldsymbol{e}_a^t)}{\tau}\right), \sigma\left(\frac{f(\boldsymbol{h}_k^{t-1}, \boldsymbol{e}_a^t)}{\tau}\right)\right),$$
$$\mathcal{L}_{KD}^t = \sum_{\mathcal{S}_u^t \in \mathcal{D}^t} \sum_{k=1}^{K_u^{t-1}} \mathcal{L}_{KD,k,u}^t, \tag{10}$$

where σ is the sigmoid function, and τ is the temperature parameter to get soft targets. Here we adopt the dot-product as the matching function f. We follow the function of distillation loss in [10], which replaces the softmax function with the sigmoid function and is efficient to compute. In this way, the new interest vectors will keep similar semantics to the existing interests even when their vector is far from the original place in high-level interest vectors' space. We call this component for existing-interests retention as *interests retainer*.

C. New-interests Detection

The previous section proposes a way to preserve the existing interests during incremental learning for MSR. However, a shared and predefined interest number lacks enough diversity and adjustability for the incremental scenario, where the existing interests and new interests will interfere with each other, e.g., the existing interests might be overlaid by new interests or prevent the new interests being captured due to the fixed interests capacity. Thus, we introduce a new-interests detector in this section which determines when to create new interests based on the distribution of items number being classified to all interests.

Given a case where a user interacts with both skirt and LEGO in one time span and the user has bought toys but no clothing-related items before, we visualize the dot-products of these two items to the existing interests and new interests. In Figure 2, we find that skirt has similar dot-products with all eight existing interests and the LEGO has the largest dot-product on the third interest. If we give a new interest vector

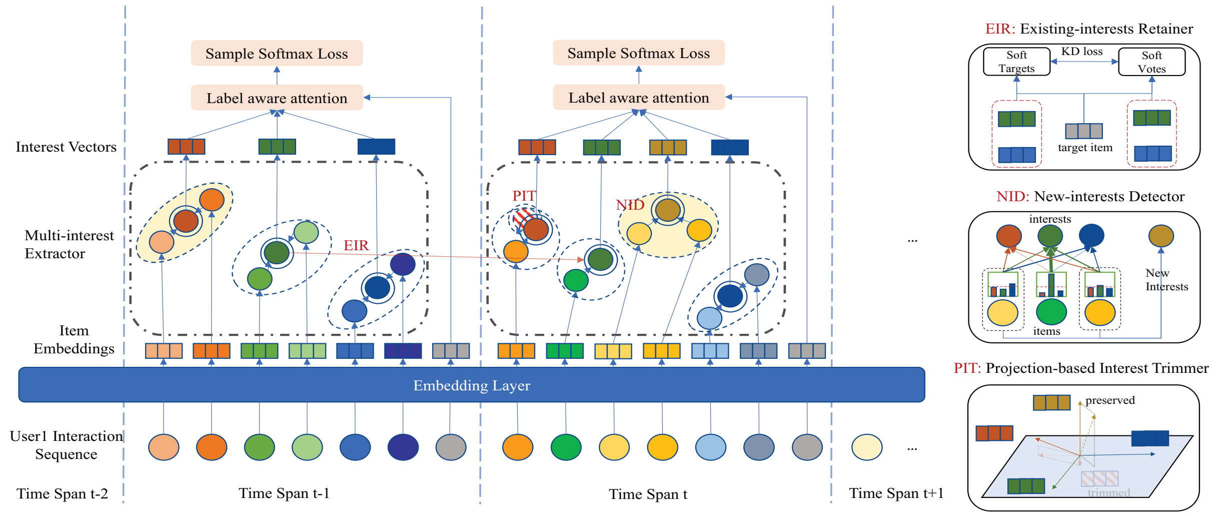
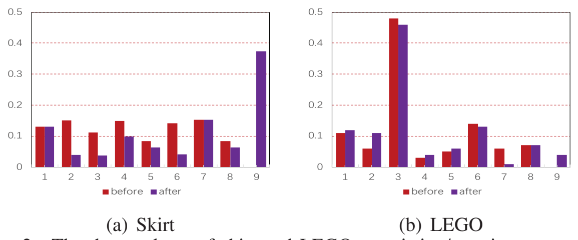


Fig. 1. The framework overview. The left part shows the fine-tuning method in time span t. The right part gives illustrations for EIR (Section IV-B), NID (Section IV-C), and PIT (Section IV-D).



(a) Skirt (b) LEGO

Fig. 2. The dot-products of skirt and LEGO to existing/new interests using DR before training (red) and after training (purple).

and retrain the model using fine-tuning strategy, the skirt has the largest dot-product on the newly created ninth interest, and LEGO keeps unchanged. We may infer that the third interest for this user is toy-related, and there are no clothing-related interests. This case illustrated that some items have similar dot-products on all the interests because they are "puzzled" and can not be classified to any of the interests.

Based on the above observation, we define the *puzzlement* of an item for multi-interest extractor and provide new interest vectors only if most of the items have a certain extent of *puzzlement*. We use Query Sparsity Measurement [22], which uses KL divergence between the query's attention probability distribution and the uniform distribution to select the most dominant dot-product pairs to represent the *puzzlement*. We do not have a query mechanism in MSR models. However, we can take the dot-product from item embedding $\boldsymbol{e}_i$ to interest vectors $\boldsymbol{h}_k$ in a probabilistic view as:

$$p(\boldsymbol{h}_k|\boldsymbol{e}_i) = \frac{\text{kernel}(\boldsymbol{e}_i|\boldsymbol{h}_k)}{\sum_{k'=1}^{K^t} \text{kernel}(\boldsymbol{e}_i|\boldsymbol{h}_{k'})}, \quad \text{kernel}(\boldsymbol{e}_i|\boldsymbol{h}_k) = e^{\boldsymbol{e}_i \boldsymbol{h}_k^\top}, \tag{11}$$

where $p(\boldsymbol{h}_k|\boldsymbol{e}_i)$ is the posterior probability of the i^{th} item being classified to the k^{th} interest. For brevity, we omit the subscript u and t. Intuitively if $p(\boldsymbol{h}_k|\boldsymbol{e}_i)$ is close to a uniform distribution $p(\boldsymbol{h}_k|\boldsymbol{e}_i) = 1/K^t$, it means the item can not be well classified into any existing interests and the new interest capsules are needed. Naturally, the distance between two distributions p and q can be used to decide when to create new interests. Here we measure the distance through the Kullback-Leibler divergence as follows:

$$KL(q||p) = \ln \sum_{k'=1}^{K^t} e^{\boldsymbol{e}_i \boldsymbol{h}_k^\top} - \frac{1}{K^t} \sum_{k'=1}^{K^t} \boldsymbol{e}_i \boldsymbol{h}_k^\top - \ln K^t. \tag{12}$$

Definition 2 (Puzzlement): Given the i^{th} item's embedding and all interest representations, the i^{th} item's *puzzlement* is defined as:

$$P(i) = \frac{1}{K^t} \sum_{k'=1}^{K^t} \boldsymbol{e}_i \boldsymbol{h}_k^\top - \ln \sum_{k'=1}^{K^t} e^{\boldsymbol{e}_i \boldsymbol{h}_k^\top} + \ln K^t, \tag{13}$$

where K^t denotes the interests number in time span t, e_i denotes the embedding of the i^{th} item and h_k denotes the representation of the k^{th} interest.

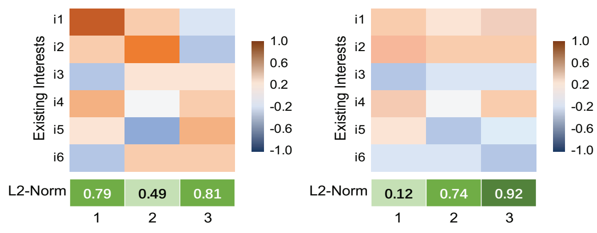
In the implementation, we will create a predefined number of new interest vectors if the average of all items' *puzzlement* for user u is larger than c_1, which is formally defined as:

$$\overline{P(i)} > c_1, i \in \mathcal{S}_u^t. \tag{14}$$

The set of all these users is the *puzzled* set $\mathcal{U}_p^t$. Here the c_1 is the hyperparameter that controls the sensitivity of the new-interests detector. We call this component for new-interests detection as *interests detector*.

D. New-interests Expansion

The previous section uses *puzzlement* to detect whether the user develops new interests. However, it is not enough because we do not know how many new interests have been developed. Intuitively, there are three ways to decide the number of new interests. The first idea is to create just one interest capsule if a new interest is detected each time. This way is easy to implement, but it is not applicable when more than one interest is developed. An improved idea is to create δK new interests certainly. This way is also easy for implementation and more applicable, but it may create too many unnecessary interests in the later time spans and become a great burden to the memory. Moreover, some new interest vectors learn redundant



(a) User1 (b) User2

Fig. 3. An illustration for two issues of new interests learned without trimming. The upper parts show the Pearson correlation coefficients between existing interests and new interests of two users. The bottom parts show the L2-norm (Euclidean norm) of new interest vectors.

existing interests or even learn nothing. Here are two example users from **Taobao** that both have 6 existing interests and 3 new learned interests by the model with fixed new-interests expansion number, as shown in Figure 3. We compute the dot-product similarities between interest k and n_u^t user's interacted item embeddings and maintain the similarity values for interest k in $\boldsymbol{p}_k = [p_{k,0}, p_{k,1}, \cdots, p_{k,n_u^t}]$. For any two interests k and j, we measure the Pearson correlation coefficient between $\boldsymbol{p}_k$ and $\boldsymbol{p}_j$, which is denoted as P_{kj}. L2-Norm denotes the Euclidean norm of each new interest vector. Higher P_{kj} means higher positive correlation and higher redundant extent for these new interests such as User1's new interest 1 and existing interest 1. And the lower L2-norm means lower existence of learned interests such as User2's new interest 1.

We design a *Projection-based Trimming* mechanism which enables the model to adjustably preserve the right number of new interest vectors for each user which correspond to the new interests developed in the new interaction sequence. The main idea is that we only preserve the orthogonal part of new interest vectors against existing interest vectors' plane and trim the new interest vectors with too small L2-norm. The reasons are two-fold. First, if a new interest vector is close to the existing interest vectors' plane, it intuitively means it represents an interest which is actually the combination of the existing interests, and we do not need to create new vectors for it. Second, the L2-norm of the vector represents the existence of the semantic this interest contains in MSR literature [5], [7], which means we can trim the vectors with too small L2-norm because they just learn trivial interests or learn similar interests against the existing ones.

Thus, we first create δK new random initialized new interest vectors for each user u at time span t. The redundant multi-interest user representation learnt by the extractor becomes:

$$\boldsymbol{H}_{u,rddt}^t = (\boldsymbol{h}_{u,1}^t, \cdots, \boldsymbol{h}_{u,K^t}^t, \boldsymbol{h}_{u,K^t+1}^t, \cdots, \boldsymbol{h}_{u,K^t+\delta K}^t), \tag{15}$$

where $\boldsymbol{H}_{u,rddt}^t \in \mathbb{R}^{d\times(K^t+\delta K)}$. δK is a hyperparameter shared for all users in all time spans which controls the interest expanding number. Thus, we employ a projection action onto $\boldsymbol{H}_{u,rddt}^t$. Specifically, we project all new interest vectors $\boldsymbol{h}_{u,K^t+1}^t, \cdots, \boldsymbol{h}_{u,K^t+\delta K}^t$ onto the existing interest vectors' plane $span(\boldsymbol{h}_{u,1}^t, \cdots, \boldsymbol{h}_{u,K^t}^t)$. The projection can be

Algorithm 1: Interests Expansion (IntsEx)

```
Input  : base model parameters $\{\boldsymbol{W}^{t-1}\}_{base}$,
         user interest vectors $\{\boldsymbol{H}_u^{t-1}\}_{u\in\mathcal{U}}$,
         incremental dataset $\mathcal{D}_t$
Output: user interest vectors$\{\boldsymbol{H}_u^t\}_{u\in\mathcal{U}}$ in time span $t$
for user $u \in U$ do
    for item $i \in \mathcal{S}_u^t$ do
        $P(i) \leftarrow$ calculate the *puzzlement* by Eq.(13);
    end
    // detect new interests
    if $\overline{P(i)} > c_1, i \in \mathcal{S}_u^t$ then
        $\delta K_u^t \leftarrow [K_u^{t-1}+1, \cdots, K_u^{t-1}+\delta K]$;
        for interest vector $k \in \delta K_u^t$ do
            initialize $\boldsymbol{h}_k^0 \leftarrow \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
        end
        $\boldsymbol{H}_u^{t-1} \leftarrow \boldsymbol{H}_u^{t-1} \cup (\boldsymbol{h}_{K_u^{t-1}+1}^{t-1}, \cdots, \boldsymbol{h}_{K_u^{t-1}+\delta K}^{t-1})$;
        $\boldsymbol{H}_u^t \leftarrow$multi-interest extraction by Eq.(4) or
         Eq.(9);
        $\boldsymbol{H}_{u,base}^t \leftarrow$ interest projection by Eq.(16);
        // trim trivial interests
        $\boldsymbol{H}_u^t \leftarrow \boldsymbol{H}_u^t \backslash \{\boldsymbol{h}_k^{t-1} \| \|\boldsymbol{h}_k^{(l),t}\|_{\text{L2}} < c_2, k \in \delta K_u^t\}$
         $K_u^t \leftarrow |\boldsymbol{H}_u^t|$
    else
        $\boldsymbol{H}_u^t \leftarrow$multi-interest extraction by Eq.(4) or
         Eq.(9);
    end
end
```

calculated as:

$$\boldsymbol{h}_{u,k}^{t,proj} = \boldsymbol{M}_{\text{exist}} \boldsymbol{M}_{\text{exist}}^{\top} (\boldsymbol{M}_{\text{exist}} \boldsymbol{M}_{\text{exist}}^{\top})^{-1} \boldsymbol{h}_k^t, \tag{16}$$

where $\boldsymbol{M}_{\text{exist}} \in \mathbb{R}^{d\times K^t}$ refers to the concatenation of $\boldsymbol{h}_{u,K^t+1}^t, \cdots, \boldsymbol{h}_{u,K^t+\delta K}^t$. In this way, only the component orthogonal to the existing interests will be preserved in the final new interest vector $\boldsymbol{h}_k^t$ after multi-interest extraction.

For the last step, we will remove the new interest vectors with trivial L2-norm using the following equation:

$$\|\boldsymbol{h}_{k,proj}^t\|_{\text{L2}} < c_2. \tag{17}$$

In this way, the *Projection-based Trimming* only preserves the orthogonal and new interests with non-trivial existence. The c_2 is a hyperparameter that controls the strictness of trivial interest trimming. We call this component for trimming the redundancy interests as *interests trimmer*. The whole interests expansion algorithm is presented in Algorithm 1.

E. Training and Inference Procedure

By integrating the *interests retainer*, *interests detector* and *interests trimmer* components together, the whole training process for incremental learning of MSR is as followed. For each user u we initialize K_u^0 interest vectors. Then we pretrain a base MSR model on all the user's historical interaction sequences. In this way, we obtain the K_u^1 interest vectors for each user, and all the historical data can be discarded. For the

Algorithm 2: The IMSR approach

```
initialize base model parameters $\{\boldsymbol{W}^0\}_{base}$;
for user $u \in U$ do
    for interest vector $k \in [1, \cdots, K_u^0]$ do
        initialize $\boldsymbol{h}_k^0 \leftarrow \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
    end
end
$\{\boldsymbol{W}^t\}_{base}$, $\{\boldsymbol{H}_u^1\}_{u \in \mathcal{U}} \leftarrow$ pretrain the base model;
for time span $t \in [2, \cdots T]$ do
    $\{\boldsymbol{W}^t\}_{base}$, $\{\boldsymbol{H}_u^t\}_{u \in \mathcal{U}} \leftarrow$ Training($\{\boldsymbol{W}^{t-1}\}_{base}$, $\{\boldsymbol{H}_u^{t-1}\}_{u \in \mathcal{U}}$);
    // inference
    calculate the $\boldsymbol{v}_u^t$ on using Eq.(5);
    for item $i \in \mathcal{S}_u^{t+1}$ do
        calculate the inner-product of $\boldsymbol{v}_u^t$ and item embedding as the item score;
    end
end
Training($\{\boldsymbol{W}^{t-1}\}_{base}$, $\{\boldsymbol{H}_u^{t-1}\}_{u \in \mathcal{U}}, \mathcal{D}_t$):
for Epoch $i \in [1, \cdots r]$ do
    // interests expansion
    $\{\boldsymbol{H}_u^t\}_{u \in \mathcal{U}} \leftarrow$
    IntsEx($\{\boldsymbol{W}^{t-1}\}_{base}$, $\{\boldsymbol{H}_u^{t-1}\}_{u \in \mathcal{U}}, \mathcal{D}_t$);
    // interests retention
    $\mathcal{L}_{SS}^t \leftarrow$get the sample softmax loss using Eq.(6);
    $\mathcal{L}_{KD}^t \leftarrow$get the knowledge distillation loss Eq.(10);
    update $\{\boldsymbol{W}^{t-1}\}_{base}$ by optimizing $\mathcal{L}_{SS}^t + \mathcal{L}_{KD}^t$;
end
$\{\boldsymbol{W}^t\}_{base} \leftarrow \{\boldsymbol{W}^{t-1}\}_{base}$;
return $\{\boldsymbol{W}^t\}_{base}$, $\{\boldsymbol{H}_u^t\}_{u \in \mathcal{U}}$;
```

time span 2, we have the new interaction sequence for each user. Then we retrain the model for r epochs. First, we use the *interests detector* on these new interactions to determine which users have new interests, and we randomly initialize δK interest vectors for each of these users. Second, we split the user's new interactions into two parts as historical sequences and target items set in time order. For all the users not detected new interests, we compute the user interest vector using multi-interest extractor directly. For others who have new interests detected, we use *interests trimmer* after the multi-interest extractor to preserve the orthogonal part of new interests and trim the new interests with trivial L2-norm. After this step, we obtain K_u^2 interests for each user u. Third, we calculate the sample softmax loss $\mathcal{L}_{SS}^2$ by Eq.(6) and the knowledge distillation loss $\mathcal{L}_{KD}^2$ using the *interests retainer*. Then we use backpropagation to update the parameters. For the next time span, the procedure is identical. In the inference procedure, we calculate the inner-product of $\boldsymbol{v}_u^t$, which is obtained by Eq.(5), and item embedding as the item score. Then we evaluate the learned MSR models in the next time span. The pseudocode for IMSR is provided in Algorithm 2.

F. Implementation Details

We implemented our proposed IMSR approach using Pytorch 1.8.1 and trained on a 64-bit Linus server equipped with 32 Intel Xeon@2.10GHz CPUs, 128GB memory, and one Titan RTX 2080ti GPU. We choose 64 as the embedding dimension of the item embeddings. The batch size is set to 128. We choose the Adam optimizer [23] to train the model and perform early stopping in the training process. The URL link to our code repository is: https://github.com/Cloudcatcher888/IMSR.

V. Experiment

To evaluate the performance of our proposed IMSR framework, we conduct extensive experiments to answer the following research questions:

- **RQ1** Can IMSR outperform the existing incremental learning strategies on multi-interest sequential recommendation?
- **RQ2** How does each component of IMSR contribute to its effectiveness?
- **RQ3** How do the hyperparameters influence the performance of IMSR?
- **RQ4** Where do the improvements of IMSR come from?

A. Experimental Setup

1) Datasets: We use four real-world sequential recommendation datasets from Amazon[1] and Taobao[2], which are adopted by many existing MSR works [5], [6].

- **Amazon**: This dataset consists of reviews of different kinds of products from Amazon [24], [25]. We consider three categories of products and obtain **Electronics**, **Clothing** and **Books** subsets. We use the item id and the UNIX review time from the metadata.
- **Taobao**: The dataset is collected from the e-commerce platform Taobao [26]. In our experiment, we only use the click behaviors and sort one user's behaviors by time.

For a fair comparison, we follow the same data preprocessing and data splitting rule for all the approaches. We discard all users with fewer than 30 interactions. The whole timeline $[0, Z]$ is split into $T+1$ time spans, where the first time span is $[0, \alpha Z]$ and $[\alpha Z, Z]$ is equally divided into T time spans. We set $T = 6$ and $\alpha = 0.5$ for all the datasets. Each user interaction is assigned to the corresponding time span. The interactions whose timestamps are in the range of $[0, \alpha Z]$ compose the pre-training dataset, while the interactions in the following T time spans are T incremental datasets. For each user in each time span, we use the latest interaction for testing, the second last interaction for validation, and all the remaining interactions for training. For the full retaining strategy, we use the pre-training dataset and $1^{th}, 2^{th}, ..., t^{th}$ incremental datasets to retrain model in the time span t. For the incremental learning strategies (including IMSR), we use the t^{th} incremental dataset to fine-tune model in the time span t. We test model on the $(t+1)^{th}$ incremental dataset. We

[1]http://jmcauley.ucsd.edu/data/amazon/

[2]https://tianchi.aliyun.com/dataset/dataDetail?dataId=649&userId=1

TABLE II
STATISTICS OF THE DATASETS.

Dataset	#users	#items	#interactions						
			pre-training	1	2	3	4	5	6
Electronics	87,912	234,621	1,689,188	224,421	428,149	329,194	129,482	481,491	196,451
Clothing	285,464	376,859	5,748,920	864,371	574,922	957,329	1,134,792	943,422	1,274,084
Books	459,133	313,966	8,898,041	1,345,234	1,324,545	1,852,324	1,593,281	1,349,281	1,433,376
Taobao	976,779	1,708,530	85,384,110	12,329,481	14,481,123	22,129,123	9,329,128	14,238,129	12,877,126

have excluded the test performance of the pretrained model (in 0^{th} time span) and computed the average performance over $1^{th}, 2^{th}, ..., (T-1)^{th}$ time spans. The statistics of the datasets are listed in Table II.

2) Evaluation: We use the hit ratio (HR), and NDCG on Top20 as the metrics to evaluate the performance of all the comparison methods, which are commonly used in multi-interest sequential recommendation [20], [27].

3) Base Models: We consider two dynamic-routing-based MSR models and one self-attention-based MSR model, which are listed as follows.

- **MIND** [5] is a typical dynamic-routing based model for multi-interest sequential recommendation. It uses a simplified transformation matrix called shared bilinear mapping matrix to extract multiple interest representations from user interaction sequences. The routing logits are initialized randomly. The default number of interest representations K_u^t is set to 4.
- **ComiRec-DR** [6] is another dynamic-routing based model for MSR. It uses a shared affine transformation matrix to extract interest representations from user interaction sequences. The routing logits are initialized as zero. The default number of interest representations K_u^t is set to 4.
- **ComiRec-SA** [6] is a self-attention based model for MSR. It employs a multi-head self-attention module to extract interest representations from user interaction sequences, where each head represents one interest. The default number of heads K_u^t is set to 4.

We choose 64 as the dimension of interest representations for all three models on all the datasets.

4) Compared Learning Strategies: We compare our proposed IMSR with the full retraining strategy and three existing incremental learning strategies on the above three base models.

- **Full retraining (FR)** In each time span, all the model parameters will be reinitialized and the whole user historical interaction sequences will be used to retrain the base model. The interests number will be kept same as IMSR .
- **Fine-tuning (FT)** In each time span, all the parameters will be inherited from the previous time span and only the newly collected user interactions will be used to fine-tune the base model.
- **SML** [28] is a model-based incremental learning approach that employs a CNN-based transfer module to leverage the previous model knowledge while training the current model on $\mathcal{D}_t$. We use 5×5 filters. The best results are reported by choosing the number of CNN filters in $\{2, 5, 8, 10\}$ and the MLP hidden size in $\{10, 20, 40, 80\}$.
- **ADER** [9] is a sample-based incremental learning approach for the session-based recommendation that samples historical interaction sessions from a session pool according to cosine similarity with the newly collected sessions as the complementary set for new sessions, which is useful to preserve the user's long-term interests. We add 5 randomly truncated interaction sequences to the session pool for each user in each time span.

For a fair comparison, all compared learning methods are optimized with the loss function in Eq. (6). The regularization coefficient for $\mathcal{L}_{KD}$ is tuned in $\{1e{-}2, 1e{-}3, \cdots, 1e{-}6, 0\}$. The learning rate is tuned in $\{0.1, 0.01, 0.005, 0, 001\}$. The incremental training epoch is tuned in $\{5, 10, 15, 20, 30, 50\}$.

B. Experimental Results: RQ1

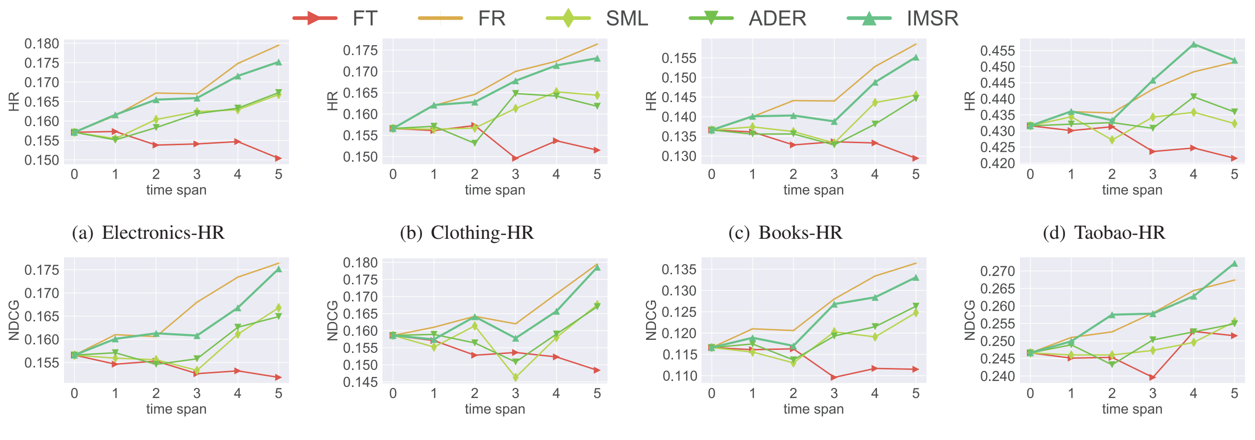
1) Performance Comparison: Table III provides the performance comparison results of different methods on four datasets.Each result is averaged by 10 repeated experiments.

We have the following observations. Table III shows SML and ADER outperform FT in most scenarios. The reason is that they both partially preserve the existing interests by sampling truncated historical interaction sequences (ADER) or transferring knowledge from previous model parameters (SML). However, their performances are inferior to FR, which leverages the whole user interaction sequences. Table III shows IMSR achieves 3.77%, 3.89%, 4.21%, 4.76% relative improvements on NDCG compared to the second best incremental learning methods on four datasets (averaged on three base models), respectively. The reason for the performance improvement is that IMSR can capture new interests with NID and PIT while SML and ADER do not expand the model capacity over time. Moreover, the performance improvements on different base models are close, which reflects that IMSR is effective in performing incremental multi-interest recommendation on different kinds of MSR models. Figure 4 gives the detailed performance trends of different methods over time spans using ComiRec-DR (similar trends can be found on other base models). The performance of FT decreases significantly over time spans. The results of SML and ADER also drop fast, which also testifies that SML and ADER cannot losslessly preserve existing interests from historical interactions. Moreover, the result of IMSR drops slightly faster than FR by only using the newly collected interactions thanks to the new-interests expansion mechanism, which alleviates the existing interests forgetting problem. Figure 4 shows that the performances of all the compared incremental learning methods become worse on **Taobao** except IMSR. The reason is that **Taobao** has more items and users' interests change more

TABLE III

THE PERFORMANCE COMPARISON RESULTS. * DENOTES $p < 0.05$ WHEN PERFORMING THE TWO-TAILED PAIRWISE T-TEST ON IMSR WITH THE INCREMENTAL LEARNING METHODS (SML OR ADER). THE **BOLD** AND THE UNDERLINE SHOW THE BEST AND SECOND-BEST RESULTS WITHIN FOUR INCREMENTAL LEARNING METHODS, RESPECTIVELY. RI INDICATES A RELATIVE IMPROVEMENT OF THE AVERAGE SCORE OF HR AND NDCG AGAINST FT. ALL THE NUMBERS IN THE TABLE ARE PERCENTAGE NUMBERS WITH '%' OMITTED.

Base model	Learning method	Electionics			Clothings			Books			Taobao		
		HR	NDCG	RI	HR	NDCG	RI	HR	NDCG	RI	HR	NDCG	RI
MIND	FR	16.03	16.43	11.15	16.23	15.98	10.57	13.82	11.95	10.47	43.29	24.90	2.63
	FT	14.75	14.46	-	14.45	14.68	-	12.34	10.98	-	42.09	24.35	-
	SML	15.41	15.17	4.71	15.27	14.81	3.26	13.12	11.12	3.97	42.88	24.58	1.54
	ADER	15.64	14.98	4.84	15.62	15.20	5.76	12.92	11.48	4.64	42.90	24.24	1.05
	IMSR	15.81*	15.71*	**7.93**	15.81*	15.71*	**8.19**	13.99*	11.94*	**11.18**	43.94*	25.66*	**4.76**
ComiRec-DR	FR	17.00	16.79	9.85	16.91	16.75	9.82	14.79	12.79	12.06	44.29	25.87	4.23
	FT	15.41	15.35	-	15.36	15.28	-	13.30	11.30	-	42.62	24.68	-
	SML	16.16	15.85	4.09	16.08	15.77	3.92	13.92	11.85	4.74	43.28	24.89	1.28
	ADER	16.12	15.90	4.10	16.02	15.84	3.96	13.73	11.96	4.43	43.44	25.00	1.68
	IMSR	16.80*	16.48*	**8.20**	16.74*	16.47*	**8.38**	14.46*	12.48*	**9.51**	44.48*	26.00*	**4.72**
ComiRec-SA	FR	17.15	16.95	10.82	16.74	16.87	8.83	14.86	12.85	11.66	44.31	25.75	4.54
	FT	15.31	15.46	-	15.49	15.39	-	13.46	11.35	-	42.44	24.58	-
	SML	15.96	15.99	3.83	15.90	15.88	2.89	13.78	11.71	2.72	43.17	24.83	1.47
	ADER	16.32	15.88	4.63	16.14	15.88	3.67	13.55	11.98	2.87	43.43	25.00	2.12
	IMSR	16.97*	16.32*	**8.19**	16.94*	16.56*	**8.45**	14.38*	12.49*	**8.30**	44.58*	26.11*	**5.48**



(a) Electronics-HR (b) Clothing-HR (c) Books-HR (d) Taobao-HR

(e) Electronics-NDCG (f) Clothing-NDCG (g) Books-NDCG (h) Taobao-NDCG

Fig. 4. HR and NDCG on each time span.

TABLE IV

THE PERFORMANCE COMPARISON RESULTS BETWEEN IMSR AND LIFE-LONG MSR MODELS. THE AVERAGE HR OVER 5 TIME SPANS ARE REPORTED.

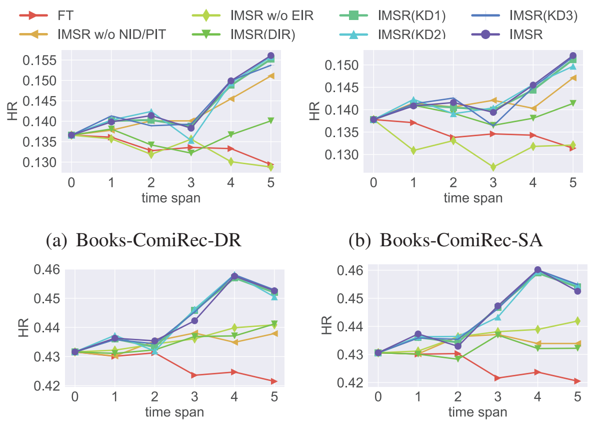
Datasets	Electronics	Clothings	Books	Taobao
MIMN	14.11	14.37	11.87	41.02
LimaRec	15.31	15.02	13.07	42.33
IMSR (ComiRec-DR)	16.81	16.68	14.48	44.35

rapidly, which will amplify the superiority of IMSR with the capability of new-interests expansion. Table IV provides the HR results between IMSR based on ComiRec-DR and life-long MSR models (includes MIMN [8] and LimaRec [7]). MIMN adopts Neural Turing Machine to adaptively read or write user interests according to the online interactions. LimaRec employs linear self-attention to identify relevant information from users' interaction sequences with different interests. These two works differ from ours in the sense that they focus on incrementally updating users' representations during online inference and do not provide an incremental learning method for MSR model updating. On average, IMSR achieves 3.61%, 2.89%, 5.12%, 4.47% relative improvements on HR compared to the best life-long MSR model (LimaRec) on the four datasets, respectively. We observe that the life-long MSR models perform inferior to IMSR because they only update user representations but do not update the model parameters after pretraining. Another reason for the performance gap is that life-long MSR models use fixed number of interests while IMSR can adaptively create vectors to capture newly evolved interests.

2) Speed-up: We compare the training time of different methods at different time spans and average inference time over all time spans. Table V shows the time cost on **Taobao**. Similar conclusions can be drawn from the other datasets. The training time of FR (on MIND and ComiRec-DR) and ADER grows linearly due to the increasing lengths of interaction sequences or the larger pool size of truncated historical interaction sequences. The training time of FR on ComiRec-SA increases more rapidly because the self-attention module

TABLE V
TRAINING/INFERENCE TIME (IN SECONDS) ON **TAOBAO** DATASET.

Base Model	Learning Method	t=1	t=2	t=3	t=4	t=5	Average Inference Time
MIND	FR	4,671	4,877	4,891	5,189	5,382	0.15
	FT	738	819	802	822	799	
	SML	908	922	941	913	902	
	ADER	982	1,154	1,379	1,534	1,712	
	IMSR	811	832	851	804	823	0.17
ComiRec-DR	FR	5,472	5,693	5,871	5,902	6,023	0.17
	FT	928	949	932	941	946	
	SML	1,052	1,098	1,079	1,073	1,081	
	ADER	990	1,199	1,499	1,591	1,891	
	IMSR	941	962	954	994	983	0.21
ComiRec-SA	FR	5,569	6,214	7,112	8,219	9,401	0.31
	FT	972	991	992	1,001	982	
	SML	1,111	1,121	1,141	1,101	1,102	
	ADER	1,054	1,231	1,529	1,681	1,952	
	IMSR	1,012	1,031	1,041	1,104	1,083	0.33



(a) Books-ComiRec-DR (b) Books-ComiRec-SA

(c) Taobao-ComiRec-DR (d) Taobao-ComiRec-SA

Fig. 5. Ablation Study on **Books** and **Taobao**.

has quadratic time complexity. The training time of SML is stable across different time spans because it does not vary the model capacity. However, SML requires longer training time in each time span due to the high computational complexity of its meta-learner. IMSR is about 6 times faster than FR (on MIND and ComiRec-DR), and the retraining time is stable across different time spans. The IMSR uses 3.5% extra training time compared with FT to achieve significant performance improvement. We find the inference time depends on both the base model and the number of interests, where IMSR takes slightly longer inference time (at the 100ms level per instance) due to the adaptive number of interests.

C. Ablation Study: RQ2

We perform an ablation study on the two largest datasets, **Books** and **Taobao**, using ComiRec-DR/SA to evaluate the effects of different components in IMSR on the recommendation performance. Specifically, we consider the following variants.

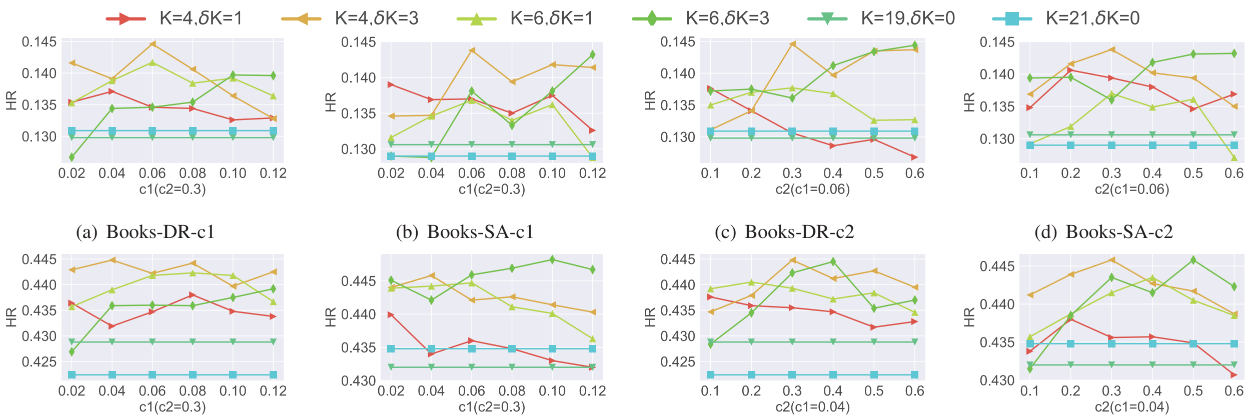
- **FT:** Using the base model with FT as the training method.
- **IMSR w/o NID&PIT:** Removing NID and PIT from IMSR.
- **IMSR w/o EIR:** Removing the EIR module from IMSR.
- **IMSR(DIR):** Replacing the EIR with the DIR for interest retention, which changes the Eq. (10) with the Euclidean distance-based regularization term [18].
- **IMSR(KD1/KD2/KD3):** Replacing Eq. (10) with three softmax-based distillation losses [21], [29], [30] to evaluate the effects of different distillation losses on the recommendation performance.

As shown in Figure 5, IMSR performs best among all the comparison methods both on DR and SA models, which shows that the removal of any component from IMSR will hurt the final performance, and the contribution of all three components is insensitive to the base model. On **Taobao**, the effectiveness of the NID and PIT is significant, which reflects that users in **Taobao** develop new interests fast due to the richness of item categories. Specifically, we observe the average interests number of all users in **Taobao** increases from 4.0 to 9.2 with IMSR. However, on **Books**, the effectiveness of the EIR is more significant, which reflects that users' interests in books are more stable, and it is more important to preserve users' existing interests. As evidence, the average interests number of all users in **Books** only increases from 4.0 to 5.6 with IMSR. On **Books**, the removal of EIR makes the performance of IMSR even inferior to FT. The performance decrease is also significant on **Taobao**. This reflects that preserving existing interests is the foundation for new-interests detection. We can see that using a distance-based regularization term for interest retention has the inferior performance to using knowledge-distillation-based regularization. We also find IMSR(DIR) performs worse on items which do not exactly match with existing interests. For instance, IMSR(DIR) fails to recommend smartphones for a user who used to be interested in flip phones, while IMSR works. The above two observations show that EIR is more flexible than DIR on existing interests preserving. Nevertheless, different kinds of knowledge distillation terms achieve similar performance, which testifies that our method is insensitive to the choice of the distillation loss function.

D. Parameter Sensitivity: RQ3

We investigate the sensitivity of the puzzlement threshold c_1 in new-interests detection and L2-norm threshold c_2 in projection-based interest trimming on IMSR performance together with different values of K for initial interests and the different values of δK for newly created interests. We choose the ComiRec-DR/SA as the base models and test on **Books** and **Taobao** two datasets. Similar conclusions can be drawn from other datasets and base models.

1) Hyperparameters c_1 and c_2: We first vary the value of c_1 in $\{0.02, 0.04, 0.06, 0.08, 0.10, 0.12\}$ and c_2 is set to 0.3 on both datasets. As shown in Figure 6, the model achieves the highest performance with moderate values of c_1 in most cases because too large c_1 prevents the creation of the new interest. Then we vary the value of c_2 in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ and c_1 is set to 0.04/0.06 on **Books** and **Taobao** respectively. In Figure 6, the model achieves the highest performance also with moderate values of c_2 in most cases because too small c_2 prevents the trivial interests from trimming.


(a) Books-DR-c1 (b) Books-SA-c1 (c) Books-DR-c2 (d) Books-SA-c2

(e) Taobao-DR-c1 (f) Taobao-SA-c1 (g) Taobao-DR-c2 (h) Taobao-SA-c2

Fig. 6. Performance with different c_1,c_2, initial K numbers and δK numbers on **Books** and **Taobao** datasets.

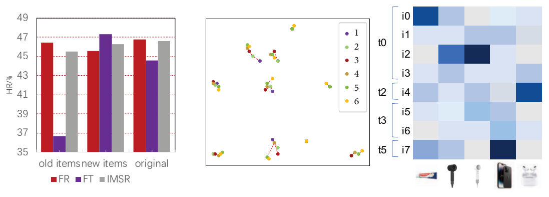

(a) (b) (c)

Fig. 7. Three case studies. (a) Performance of FR, FT and IMSR of 5^{th} time span in **Taobao** grouped by three item types: only existing items, only new items and the original datasets. (b) t-SNE visualization of one user's interest evolution among different time spans. (c) Heatmap of dot-products between each interest $i(j)$ created in time span $t(i)$ of a user and target items in the last time span.

2) Interests Number K and δK: The value of K and δK are chosen from $\{(4,1),(4,3),(6,1),(6,3),(19,0),(21,0)\}$, where $K=19,\delta K=0$ and $K=21,\delta K=0$ equal to the settings where we create all the interest vectors in advance at the pre-training stage for IMSR with $K=4,\delta K=3$ and $K=6,\delta K=3$, respectively. As shown in Figure 6, IMSR with $\delta K=3$ reports higher results than $\delta K=1$, which shows user may develop multiple interests in a new time span. IMSR achieves higher HR values when $K=6$ on **Taobao**, probably because users generally have more interests in **Taobao**. We also find that the highest performance for $K=4$ and $\delta K=1$ is achieved at smaller c_1 or c_2 probably because K, δK are relatively small and we need more loose interest expansion pattern controlled by c_1 and c_2. The performance of IMSR with $K=19,\delta K=0$ and $K=21,\delta K=0$ is far below the performance of IMSR with $K=4/6,\delta K=1/3$, which confirms the effectiveness of using interests expansion strategy and shows that the model suffer when we create too many interest vectors in advance.

E. Case Study: RQ4

Finally, we perform three case studies to illustrate the advantages of IMSR compared with FR and FT, and visualize user's interest vectors to interpret their rationality. We use ComiRec-DR model for demonstration.

1) Performance Difference for New/Existing Items: We divide the items of one time span in **Taobao** into two types: **existing item** includes the items that the user has interacted with in previous time spans; **new item** is an item the corresponding user newly interacts with in this time span. The two treatment groups are: only testing on new items; only testing on existing items. The original group is the control group. We can see that FR performs better on existing items because it uses all existing items for retraining. FT heavily forgets the existing items but performs best on new items. IMSR compromises between the preservation of the existing items and new items detection, which achieves convincing performance in both treatment groups.

2) Visualization of User's Interest Vectors: We sample one user and visualize his/her interest vector evolution among the six time spans of **Taobao** in Figure 7(b) by t-SNE. Different colors correspond to different time spans. In time span 0 (the purple scalars), the user has only 4 interests. In time span 1 (the light green scalars), the user generates 3 new interests in new places, and the 4 existing interests remain in their original places. In later time spans, We can also see that different interests in the same time span are mostly located in different places, which reflects the effectiveness of PIT and NID in preventing learning redundant interests. Vectors of the same interest in different time spans linked with red dashes locate quite closely, which shows that EIR prevents the preserved interests from drifting from the original places dramatically. We also provide a case study to show the necessity of retaining all existing interests discussed in the introduction. Typical MSR models (described in Section III) calculate the dot-product similarity as an attention score between each interest vector $\boldsymbol{h}_k$ and target item's embedding $\boldsymbol{e}_a$, and perform weighted sum over all interests as the final user representation. To this end, MSR models can adaptively assign importance to interests for recommendation. The higher attention score between interest $\boldsymbol{h}_k$ and target item's embedding $\boldsymbol{e}_a$ means the higher importance of interest k for recommending target item i_a. Among all the attention scores over interests and target items in the last time span for each user in the **Taobao** dataset calculated by ComiRec-DR, we find more than 50%/60% users bought items which have the highest attention scores with the interests developed in the first/second time span. Hence, it is beneficial to retain all existing interests including early ones. Figure 1 presents the heatmap of one user where some early interests still have high attention scores with the target items in the last time span, which shows early interests are also valuable for recommending the items in later time spans.

VI. Related Work

A. Multi-interest Sequential Recommendation

In recent years, there is a growing number of works researching personalized recommendations [1], [2], [4], [31]–[34]. One species of them typically attempts to learn user representations from historical user interaction sequences, which is referred to as sequential recommendation.

Recently, some researchers have utilized capsule networks or self-attention to capture users' diverse interests to solve this problem. The capsule network [19] was firstly used for image classification. It uses vectorized capsules to replace scalar neurons in neural networks and employs the dynamic routing mechanism based on a similarity metric to form capsules. Inspired by the high expressiveness of capsule networks, MIND [5] performs dynamic routing to extract users' high-level multi-interest capsules from the raw user interaction sequence for item recommendation. ComiRec [6] implement both capsule-network-based and self-attention-based sequence recommendation model, which optimizes the dynamic routing process in MIND and introduces a controllable method for interest selection. Unfortunately, existing capsule-network-based methods cannot preserve the sequential order among interactions and lack the ability to utilize temporal information for sequential recommendation.

Note that a recent work called LimaRec [7] employs linear self-attention to identify relevant information from users' interaction sequences with different interests. This work differs from ours in the sense that LimaRec focuses on incrementally updating users' representations for online inference. LimaRec does not provide an incremental strategy for MSR model updating and still needs to fully retrain the model using the entire historical interaction sequence periodically.

B. Incremental Learning for Recommendation

Practical recommender systems need to periodically retrain recommendation models. It is often desirable to retrain the model on both historical and new interaction data to capture long-term and short-term user interests. However, full retraining would be very time-consuming and incur high memory cost, especially when the scale of historical data is large. To address the problem, a more cost-effective way is to perform incremental learning with new interactions in each time span [35]–[37]. The ultimate goal of performing incremental learning is to update model parameters only with newly collected interactions and still achieve competitive performance as compared with full retraining. The main challenge of incremental learning is to solve the catastrophic forgetting problem, which means the model will perform worse on existing data samples after training on new datasets. There are two groups of incremental methods: model-based methods and sample-based methods. The model-based methods' main idea is to add some regularization on the loss function of new tasks to protect the existing knowledge from forgetting. Most of these kinds of methods do not require existing datasets for the model to review. The sample-based method focuses on which part of existing datasets should be preserved and how to combine the existing and new datasets for model training. In the recommendation literature, several methods have been proposed for incremental model training [38], [39]. For instance, MAN [11] uses a memory augmented neural model for the incremental session-based recommendation, which memorizes a subset of testing data to enrich the training datasets on the interfere stage. However, among all these incremental methods for recommendation, there are no specialized incremental strategies for sequential recommendation.

Another group of works focus on incrementally updating users' representations based on real-time interactions during online inference. Zhou et al. [40] proposed an expansion technique on each user's interests to generate diverse recommendation results based on streaming interaction data. Zhou et al. [41] introduced an algorithm for efficiently performing social updates in dynamic shared communities by checking the status of each sub-community and its interactions with other sub-communities. The above two works both need to fully retrain the model using all historical interactions periodically. Different from these works, we focus on updating both the interests number and model parameters using the newly collected interaction data during offline training.

VII. Conclusion

In this work, we show how the existing multi-interest sequential recommender system can be deployed in incremental scenarios by using fine-tuning strategies. Furthermore, we propose an incremental learning framework for multi-interest sequential recommendation named IMSR, which augments the base dynamic routing or self-attention-based MSR models with the existing-interests retainer (EIR), new-interests detector (NID), and projection-based interests trimmer (PIT) to alleviate the existing-interests forgetting problem and adaptively increase the number of the interests. Extensive experiments verify the effectiveness of the proposed IMSR on four real-world datasets, compared with the baseline methods.

Acknowledgement

The authors would like to thank the anonymous reviewers for their insightful reviews. This work is supported by the National Key Research and Development Program of China (2022YFE0200500), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), the Tencent Wechat Rhino-Bird Focused Research Program, and SJTU Global Strategic Partnership Fund (2021 SJTU-HKUST).

REFERENCES

[1] J. He, J. Qi, and K. Ramamohanarao, "A Joint Context-Aware Embedding for Trip Recommendations," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, vol. 00, 2019, pp. 292–303.

[2] Y. Chang, C. Zhai, Y. Liu, Y. Maarek, J. Tang, and K. Wang, "Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding," *Arxiv*, pp. 565–573, 2018.

[3] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep Interest Evolution Network for Click-Through Rate Prediction," in *AAAI*, 2018, pp. 5941–5948.

[4] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based Recommendations with Recurrent Neural Networks," *arXiv*, 2015.

[5] C. Li, Z. Liu, M. Wu, Y. Xu, P. Huang, H. Zhao, G. Kang, Q. Chen, W. Li, and D. L. Lee, "Multi-Interest Network with Dynamic Routing for Recommendation at Tmall," in *Proceedings of the 28th ACM International Conference on Information & Knowledge Management*, 2019.

[6] Y. Cen, J. Zhang, X. Zou, C. Zhou, H. Yang, and J. Tang, "Controllable Multi-Interest Framework for Recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[7] Y. Wu, L. Yin, D. Lian, M. Yin, N. Z. Gong, J. Zhou, and H. Yang, "Rethinking Lifelong Sequential Recommendation with Incremental Multi-Interest Attention," *arXiv*, 2021.

[8] A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, G. Karypis, Q. Pi, W. Bian, G. Zhou, X. Zhu, and K. Gai, "Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 7 2019, pp. 2671–2679.

[9] F. Mi, X. Lin, and B. Faltings, "ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation," *arXiv*, 2020.

[10] Y. Wang, H. Guo, R. Tang, Z. Liu, and X. He, "A Practical Incremental Method to Train Deep CTR Models," *arXiv*, 2020.

[11] F. Mi and B. Faltings, "Memory Augmented Neural Model for Incremental Session-based Recommendation," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020, pp. 2169–2176.

[12] J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J.-R. Wen, Y. Liu, Y. Zhang, F. Feng, C. Wang, X. He, M. Wang, Y. Li, and Y. Zhang, "How to Retrain Recommender System? A Sequential Meta-Learning Method," *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1479–1488, 2020.

[13] D. Peng, S. J. Pan, J. Zhang, and A. Zeng, "Learning an Adaptive Meta Model-Generator for Incrementally Updating Recommender Systems," pp. 411–421, 2021.

[14] C. Simon, P. Koniusz, R. Nock, and M. Harandi, "Adaptive Subspaces for Few-Shot Learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 00, 2020, pp. 4135–4144.

[15] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive Neural Networks," *arXiv*, 2016.

[16] T. Chen, I. Goodfellow, and J. Shlens, "Net2Net: Accelerating Learning via Knowledge Transfer," *arXiv*, 2015.

[17] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert Gate: Lifelong Learning with a Network of Experts," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7120–7129.

[18] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv*, 2015.

[19] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic Routing Between Capsules," *arXiv*, 2017.

[20] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer," in *Proceedings of the 28th ACM International Conference on Information & Knowledge Management*, 2019.

[21] Z. Li and D. Hoiem, "Learning without Forgetting," *arXiv*, 2016.

[22] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," *arXiv*, 2020.

[23] J. L. B. Diederik P. Kingma, "Adam: A Method for Stochastic Optimization," in *ICLR*, 2015, pp. 785–794.

[24] J. McAuley, C. Targett, Q. Shi, and A. v. d. Hengel, "Image-Based Recommendations on Styles and Substitutes," pp. 43–52, 2015.

[25] J. Bourdeau, J. A. Hendler, R. N. Nkambou, I. Horrocks, B. Y. Zhao, R. He, and J. McAuley, "Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering," *WWW*, pp. 507–517, 2016.

[26] Y. Guo, F. Farooq, H. Zhu, X. Li, P. Zhang, G. Li, J. He, H. Li, and K. Gai, "Learning Tree-based Deep Model for Recommender Systems," *KDD*, pp. 1079–1088, 2018.

[27] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks," *RecSys*, 2017.

[28] J. Huang, Y. Chang, and X. Cheng, "How to Retrain Recommender System? A Sequential Meta-Learning Method," *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1479–1488, 2020.

[29] A. Cheraghian, S. Rahman, P. Fang, S. K. Roy, L. Petersson, and M. Harandi, "Semantic-aware Knowledge Distillation for Few-Shot Class-Incremental Learning," *arXiv*, 2021.

[30] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large Scale Incremental Learning," *arXiv*, 2019.

[31] J. Zhang, C. Gao, D. Jin, and Y. Li, "Group-Buying Recommendation for Social E-Commerce," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, vol. 00, 2021, pp. 1536–1547.

[32] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," *WWW*, pp. 811–820, 2010.

[33] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng, "Learning Hierarchical Representation Model for NextBasket Recommendation," *SIGIR*, pp. 403–412, 2015.

[34] R. He and J. McAuley, "Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation," *arXiv*, 2016.

[35] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[36] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring Catastrophic Forgetting in Neural Networks," in *AAAI*, 2017.

[37] X. Liu, M. Masana, L. Herranz, J. V. d. Weijer, A. M. Lopez, and A. D. Bagdanov, "Rotate your Networks: Better Weight Consolidation and Less Catastrophic Forgetting," *arXiv*, 2018.

[38] Y. Tang, K. Guo, R. Zhang, T. Xu, J. Ma, and T. Chi, "ICFR: An effective incremental collaborative filtering based recommendation architecture for personalized websites," in *World Wide Web*, vol. 23, no. 2, 2020, pp. 1319–1340.

[39] F. Yuan, G. Zhang, A. Karatzoglou, J. Jose, B. Kong, and Y. Li, "One Person, One Model, One World: Learning Continual User Representation without Forgetting," in *Proceedings of the 43nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[40] X. Zhou, L. Chen, Y. Zhang, L. Cao, G. Huang, and C. Wang, "Online Video Recommendation in Sharing Community," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 5 2015, pp. 1645–1656.

[41] X. Zhou, D. Qin, X. Lu, L. Chen, and Y. Zhang, "Online social media recommendation over streams," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 938–949.