Who Will Purchase this Item Next? Reverse Next Period Recommendation in Grocery Shopping

MING LI, AIRLab, University of Amsterdam, The Netherlands
MOZHDEH ARIANNEZHAD, AIRLab, University of Amsterdam, The Netherlands
ANDREW YATES, University of Amsterdam, The Netherlands
MAARTEN DE RIJKE, University of Amsterdam, The Netherlands

Recommender systems have become an essential instrument to connect people to the items that they need. Online grocery shopping is one scenario where this is very clear. So-called *user-centered* recommendations take a user as input and suggest items based on the user's preferences. Such user-centered recommendations have received significant attention and uptake. Instead, we focus on an *item-centered* recommendation task, again in the grocery shopping scenario. In the *reverse next-period recommendation* (RNPR) task, we are given an item and have to identify potential users who would like to consume it in the next period.

We consider three sub-tasks of the overall RNPR task, (i) Expl-RNPR, (ii) Rep-RNPR, and (iii) Mixed-RNPR, where we consider different types of target users, i.e., (i) explore users, who are new to a given item, (ii) repeat users, who previously purchased a given item, and (iii) both explore users and repeat users. To address the Expl-RNPR task, we propose a habit-interest fusion model that employs frequency information to capture the repetition-exploration habits of users and that uses pre-trained item embeddings to model the user's interests. For the Mixed-RNPR task, we propose a repetition-exploration user ranking algorithm to decouple the repetition and exploration task, and investigate the trade-off between targeting different types of users for a given item. Furthermore, to reduce the computational cost at inference, we analyze the repetition behavior from both user and item perspectives and then introduce a repetition-based candidate filtering method for each sub-task. We conduct experiments on two public grocery shopping datasets. Our experimental results not only demonstrate the difference between repetition and exploration, but also the effectiveness of the proposed methods.

CCS Concepts: • **Information systems → Recommender systems**.

Additional Key Words and Phrases: Reverse next period recommendation, item-centered recommendation, repetition-exploration

1 INTRODUCTION

Recommendation systems are an important instrument to connect users and items in many online services, like e-commerce [6, 23, 52], grocery shopping [42], music/movie streaming platforms [7, 9], and news [27, 43]. Unlike the top-*n* recommendation scenario, where the assumption is that there is no temporal information about past interactions [10, 19, 50], sequential recommendation systems keep track of users' historical interactions. This allows the recommender system to model users' preferences over time and recommend items for their next interactions [45]. Various types of sequential recommendation tasks have been well investigated in recent years, such as next-item recommendation [13, 49] and next-basket recommendation [17, 22, 31, 32, 53, 54]. What unites

Authors' addresses: Ming Li, AIRLab, University of Amsterdam, Science Park 900, Amsterdam, The Netherlands, m.li@uva.nl; Mozhdeh Ariannezhad, AIRLab, University of Amsterdam, Science Park 900, Amsterdam, The Netherlands, m.ariannezhad@uva.nl; Andrew Yates, University of Amsterdam, Science Park 900, Amsterdam, The Netherlands, a.c.yates@uva.nl; Maarten de Rijke, University of Amsterdam, Science Park 900, Amsterdam, The Netherlands, m.derijke@uva.nl.

2770-6699/2023/5-ART
https://doi.org/10.1145/3595384

these tasks is their *user-centered* focus: given a user and their profile, these tasks aim to suggest relevant items that meet the user's preferences.

1.1 Item-centered recommendations

In this paper we focus on a less studied *item-centered* task, where the recommender system is given an item and needs to identify users who are most likely to consume it. Examples of such item-centered tasks emerge when advertising products [37], reducing waste [51], or promoting a healthy lifestyle [34]. For example, if a supermarket wants to sell bread that will expire soon to reduce waste, simply recommending the bread to all users will not only lead to high service costs, but it will also harm users' experience for those who do not like bread. As a result, item-centered recommendation algorithms have to identify specific *top-k* users who have an interest and may consume a given item. We define a novel "item-centered" recommendation problem in a sequential setting, namely *reverse next-period recommendation* (RNPR):[1]

> Given an item and historical transactions of all users, the reverse next-period recommendation task is to find potential users who have an interest in the item in the next time period.

Somewhat related to our item-centered focus, Wang et al. [46] have recently formulated the task of selecting potential "adopters" for a free-trial item to increase the exposure of the long-tail items. However, despite the similarity to our item-centered task, Wang et al. still focus on user-side performance.

While some previous studies have considered item-centered recommendation problems, their focus has typically been on improving efficiency in this setting rather than designing recommendation algorithms tailored to the new item-centered setting. Two related approaches that focus on efficiency are reverse maximum inner product search [reverse k-MIPS, 1] and reverse top-k queries [40, 41, 56]. These approaches do not consider the temporal information and assume that the user and item representation vectors are known or can be pre-computed in advance. As a result, they are not able to handle the RNPR task effectively.

In this paper, we are specifically interested in the grocery shopping scenario, where historical interactions consist of baskets (multi-sets of items), for the following reasons: (i) the demand for item-centered recommendation is very clear in the grocery shopping scenario, e.g., to help reduce food waste[2] or to promote healthy lifestyles;[3] (ii) repetition behavior and exploration behavior both appear in the grocery shopping scenario, which allows us to understand the imbalance between repetition and exploration in the item-centered recommendation scenario. Specifically, we regard the next n baskets after the historical interactions to be "the next time period." We consider three key aspects of the RNPR task in this work: (i) user-centered methods for item-centered tasks, (ii) repetition vs. exploration behavior of users, and (iii) efficiency.

1.1.1 *User-centered methods for item-centered tasks.*

Various sequential recommendation algorithms [11, 17, 53, 54] have been proposed and shown to achieve good performance in user-centered sequential recommendation. Even though these models are user-centered, they can also be adapted to the RNPR task, for instance by computing scores from the user's side and ranking from the item's side. An intuitive solution to adapt these models for RNPR task is that, for a given item, the model computes a score for every candidate user that reflects the user's preference for it, and then selects the top-k highest-scoring users.

Before we design task-specific solutions for RNPR, it is of interest to answer the following question: what are the performance and limitations of these user-centered sequential recommendation methods in an item-centered RNPR setting? Since we focus on the grocery shopping scenario, we assess and investigate the performance of several representative next-basket recommendation algorithms [17, 54] on the RNPR task, and we find that the

[1]While conventional recommendation systems concentrate on recommending items to users, the term "reverse" marks a shift in focus by recommending users to items.
[2]https://www.aholddelhaize.com/sustainability/our-position-on-societal-and-environmental-topics/food-waste/
[3]https://www.aholddelhaize.com/en/sustainability/healthier-choices/

performance of state-of-the-art NBR methods does not always generalize to the RNPR, even though they do model the temporal dependencies present in sequential recommendation (Section 8.1).

1.1.2 Repetition vs. exploration. In a user-centered sequential recommendation scenario, namely next-basket recommendation, a recent study [25] separates the candidate items into *repeat items* for a user, that is, items that the user has interacted with before, and *explore items* for a user, which are items that are new for the user. Similarly, for the RNPR task, given an item, we can also split its candidate users into *repeat users*, who have previously interacted with the given item, and *explore users*, who have never interacted with the given item before. We consider three sub-tasks of the RNPR problem:

Expl-RNPR: find possible new users (i.e., *explore users*), who will purchase the given item in the next period;
Rep-RNPR: find possible repeat users (i.e., *repeat users*), who will repurchase the given item in the next period; and
Mixed-RNPR: find all possible users (i.e., both *repeat users* and *explore users*), who will purchase the given item in the next period.

To address the Expl-RNPR task, we propose a *habit-interest fusion* (HIF) model that uses pre-trained embeddings to model a user's interests and employs frequency information to capture the repetition-exploration habits of the user. To train HIF effectively, we use an item-wise pairwise ranking loss and propose two strategies to construct the training samples: positive augmentation and negative adjustment. To address the Rep-RNPR task, we employ a simple time-aware frequency method, which only leverages users' direct interactions with a given item. To address the Mixed-RNPR task, we introduce a *repetition-exploration user ranking* (REUR) algorithm, which decouples repetition, i.e., recommending users who have purchased the given item before, from exploration, i.e., recommending users who have not purchased the given item, and then tries to find the optimal combination of *repeat users* and *explore users*. Importantly, REUR allows us to investigate the trade-off between recommending repeat users and explore users. We find that recommending repeat users for a given item is much easier than finding potential explore users for a given item (Section 8.4).

1.1.3 Efficiency. Real world e-commerce applications usually have a large number of users and items [1], making it computationally expensive to compute every single user's score for a given item in order to identify the top-k users. In addition, an item-centered recommender system needs to operate in an *ad hoc* fashion, where it is not known up front which item needs to be recommended [1]. Therefore, it is important to reduce the computational costs of RNPR. To this end, we propose repetition-based methods to reduce the number of candidate users for a given item. Specifically, we first analyze the statistics of users' repetition behavior on both item and category level, from both the item and user perspective, and then propose two *repetition-rule based candidate filtering methods* (RRBF), which select candidate users for a given item based on users' item level (RRBF-item) and category level (RRBF-cat) repetition behaviors. For the Expl-RNPR task, we propose a *candidate filtering model* (CFM) to predict whether a user will purchase a specific category in the next period based on the temporal category information, which can further reduce the computational costs on top of RRBF-cat. We find that both the rule-based method (RRBF) and the model-based method (CFM) can effectively reduce the computational costs of RNPR (Section 8.5).

1.2 Main contributions

The main contributions of our paper are as follows:

- We define and investigate the problem of reverse next-period recommendation (RNPR), introducing the Expl-RNPR, Rep-RNPR and Mixed-RNPR sub-tasks that consider different types of users, i.e., *repeat users* and *explore users*. To the best of our knowledge, this is the first work to study this problem.

- We investigate several sequential NBR recommendation algorithms applied to the RNPR problem, and find that their performance cannot be generalized in some cases for the Expl-RNPR task, and that they are more complex than needed for the Mixed-RNPR task.
- For the Expl-RNPR task, we propose a habit-interest fusion (HIF) model to capture users' habits and interests w.r.t. a given item, and we propose two training sample construction strategies for HIF.
- For the Mixed-RNPR task, we propose a REUR algorithm to decouple the repetition task and exploration task; and we investigate the trade-off between repetition and exploration via the REUR algorithm.
- We analyze users' repetition behavior on different levels from both a user and item perspective, and propose several repetition-based user candidate filtering methods to reduce the computational cost at inference time.
- We conduct experiments on two publicly available grocery shopping datasets, i.e., Dunnhumby and Instacart. The results demonstrate the effectiveness of the strategies we propose in this paper.

The remainder of the paper is organized as follows. We discuss related work in Section 2. Section 3 is devoted to formulated the reverse next-period recommendation problem. In Section 4 we analyze users' repeat behavior. Then, in Section 5 we introduce the habit-interest fusion (HIF) for addressing the RNPR task. Methods for candidate filtering are introduced in Section 6. Our experimental setup is described in Section 7 and our experimental results in Section 8. We conclude in Section 9.

2 RELATED WORK

2.1 User-centered recommendation

Sequential item recommendation tasks have been investigated for many years. The purpose of such tasks is to consider users and their preferences and to recommend the next item according to those preferences. Recurrent neural networks (RNNs) [8, 15] and transformers [39] have shown strong performance in modeling sequential information, and they have been widely used to learn representations of historical behavior in session-based recommendation. GRU4Rec [14] leverages GRUs to model user sequences and then optimize a ranking-based loss for session-based recommendation. An updated version, GRURec+ [13], has a new ranking loss and sampling strategy. NARM [24] couples a GRU with an attention mechanism to make the recommendation model focus more on recent baskets. SASRec [20] employs a self-attention-based method to capture the temporal dynamics of sequential recommendations in an efficient way.

In addition to RNN- and transformer-based models, several deep learning techniques have also been applied to this area. Memory networks are applied by STAMP [28] to capture a user's general interests and current interests. SR-GNN [49] models a session sequence as a graph and then uses a graph neural network [33] to capture item transactions and learn an accurate item embedding. Tang and Wang [38] propose a CNN-based method to capture general interests and sequential patterns via vertical and horizontal filtering. Yuan et al. [55] introduce a generative model to improve the performance. Pre-trained models (such as BERT) [35] and knowledge graphs are also being applied to user-centered recommendations [18, 48].

In grocery shopping, both the sequences of historical interactions and the output of recommendations are sets (or rather, multisets) of items, so-called baskets, and the *next-basket recommendation* (NBR) task is a user-centered sequential recommendation task that caters to this scenario. Over the years, many dedicated NBR methods have been proposed [25]. These include Markov chain (MC)-based methods [32, 44], deep learning-based methods [2, 4, 16, 22, 47, 53, 54], and frequency neighbor-based methods [11, 17]. An analysis conducted by Li et al. [25] assesses and evaluates the NBR performance from a new repetition and exploration perspective; their comparisons show that recommending *repeat items* (items that a user has interacted with previously) is an easier task than recommending *explore items* (items that a user has never interacted with before).

All of the sequential recommendation methods mentioned above focus on the user perspective, whereas we propose the reverse next-period recommendation (RNPR) problem that focuses on the item perspective.

2.2 Item-centered recommendations

Item-centered recommendations focus on the item perspective. That is, they aim to recommend suitable users for a given item that are likely to interact positively with it (i.e., purchase it in a grocery shopping setting, listen to it in a music recommendation setting, download and read it in a book recommendation setting, etc.). Early proposals of item-centered recommendation date back at least to the so-called reverse top-k query problem [40, 41, 56]. Early publications on this problem typically consider Euclidean spaces with low-dimensional (often, around 5) user and item vectors.

In recent years, recommender systems have benefited from the development of deep learning techniques, which can construct high-dimensional representations and embeddings of users and items. For example, Amagata and Hara [1] propose reverse top-k maximum inner product search (reverse k-MIPS), which assumes that d-dimensional representations of users and items are obtained via matrix factorization [21]. Interestingly, previous work on item-centered recommendations only focuses on efficiency (i.e., on reducing the computational costs) rather than on improving the performance on the item-centered recommendation task. Furthermore, they do not consider temporal dependencies between historical items, which is a key aspect of the sequential recommendation task. Recently, Wang et al. [46] have formulated a user selection problem for free-trial items, which aims to increase item exposure and retain user-side performance.

Unlike previous work, we formulate the RNPR problem in a sequential setting. We aim to find users who will purchase a given item and focus on item-side performance. Moreover, we do not only focus on the efficiency aspect, but also try to improve the performance on the RNPR task.

3 PROBLEM FORMULATION

In this section, we describe two types of users, i.e., *repeat users* and *explore users*, formalize the reverse next-period recommendation task, and associate three sub-tasks with it, i.e., Expl-RNPR, Rep-RNPR and Mixed-RNPR. We also introduce the candidate filtering task for reverse next-period recommendation. The notation used in this paper is shown in Table 1.

3.1 Reverse next-period recommendation

Assume we have a set of users and a set of items, denoted as $U = \{u_1, u_2, \ldots, u_o\}$ and $I = \{i_1, i_2, \ldots, i_m\}$, respectively. Each item belongs to a category $c \in C = \{c_1, c_2, \ldots, c_q\}$. B_u^t denotes user u's basket at time step t, where B_u^t consists of a set of items $i \in I$. $S_u^h = \{B_u^1, B_u^2, \ldots, B_u^t\}$ represents the sequence of historical interactions for user u, and $S_u^n = \{B_u^{t+1}, B_u^{t+2}, \ldots, B_u^{t+n}\}$ represents the sequence of the next n interactions for user u. Then, $I_u^h = \{i_1, i_2, \ldots, i_z\}$ and $I_u^n = \{i_1, i_2, \ldots, i_e\}$ represent the item set that user u has purchased before and will purchase in the next n baskets, i.e., next period, respectively. $C_u^h = \{c_1, c_2, \ldots, c_v\}$ represents the category set in which user u has purchased items before, $C_u^n = \{c_1, c_2, \ldots, c_w\}$ represents the category set in which user u will purchase items in the next n baskets.

Given a specific item i, the users in U can be divided into *repeat users* and *explore users* based on the historical interaction with the item i:

Repeat users U_i^{rep} for item i are the users u who have purchased the item i before, that is, users u such that $i \in I_u^h$.

Explore users U_i^{expl} for item i are the users u who have not purchased product i before, that is, users u such that $i \notin I_u^h$.

Table 1. Notation used in the paper.

Symbol	Description
U	Set of all users, i.e., $U = \{u_1, u_2, \ldots, u_o\}$.
I	Set of all items, i.e., $I = \{i_1, i_2, \ldots, i_m\}$
C	Set of all categories, i.e., $C = \{c_1, c_2, \ldots, c_q\}$
I^c	Set of all items belongs to category c, i.e., a subset of I.
B_u^t	t-th basket purchased by user u at time t, which is a set of items $i \in I$
S_u^h	Sequence of historical baskets for user u, i.e., $S_u^h = \{B_u^1, B_u^2, \ldots, B_u^t\}$
S_u^n	Sequence of future (next-period) baskets for user u, i.e., $S_u^n = \{B_u^{t+1}, B_u^{t+2}, \ldots, B_u^{t+n}\}$
I_u^h	Set of historical items purchased by user u
I_u^n	Set of items that user u will purchase during next period (n baskets)
C_u^h	Set of categories from which user u has purchased items before
C_u^n	Set of categories from which user u will purchase items during next period (n baskets)
U_i^{rep}	Set of *repeat users* u_i^{rep} who have purchased item i, i.e., $i \in I^h_{u_i^{rep}}$
U_i^{expl}	Set of *explore users* u_i^{expl} who have not purchased item i, i.e., $i \notin I^h_{u_i^{expl}}$
U_c^{rep}	Set of *repeat users* u_c^{rep} who have purchased an item in category c, i.e., $c \in C^h_{u_c^{rep}}$
U_c^{expl}	Set of *explore users* u_c^{expl} who have not purchased an item in category c, i.e., $c \notin C^h_{u_c^{expl}}$
U_i^t	Set of the target users for item i
$\hat{U}_i^t$	Set of the candidate users for item i
T_i	Set of ground-truth users u who will purchase item i in next period, i.e., $i \in I_u^n$
T_i^{rep}	Set of ground-truth *repeat users* $u_i^{rep,*}$ for item i, i.e., $i \in I^n_{u_i^{rep,*}}$ and $i \in I^h_{u_i^{rep,*}}$
T_i^{expl}	Set of ground-truth *explore users* $u_i^{expl,*}$ for item i, i.e., $i \in I^n_{u_i^{expl,*}}$ and $i \notin I^h_{u_i^{rep,*}}$
P_i^n	Predicted top-k users for item i, i.e., $P_i^n = [u_1^p, u_2^p, \ldots, u_k^p]$
$f(\cdot)$	Reverse next period recommendation (RNPR) algorithm
$g(\cdot)$	Candidate filtering algorithm

Similarly, given a specific category c, the users in U can also be divided as follows:

Repeat users U_c^{rep} for category c are the users u who have purchased an item in category c before, that is, users u such that $c \in C_u^h$.

Explore users U_c^{expl} for category c are the users u who have not purchased category an item in c before, that is, users u such that $c \notin C_u^h$.

Given a specific item i and historical interactions $S^h = \{S_1^h, S_2^h, \ldots, S_m^h\}$ of target users $u_1, \ldots, u_m \in U_i^t$, the goal of the *reverse next-period recommendation* (RNPR) task is to predict the top-k users $P_i^n \subseteq U_i^t$, who will purchase the given item i in one of the next n baskets. To address the RNPR task, we seek to define a function f that takes item i and historical interactions $S^h = \{S_1^h, S_2^h, \ldots, S_m^h\}$ of target users $u_1, \ldots, u_m$ as input, and returns P_i^n:

$$P_i^n = [u_1^p, u_2^p, \ldots, u_k^p] = f(i, \{S_1^h, S_2^h, \ldots, S_m^h\}). \tag{1}$$

where P_i^n is a predicted ranked list, which contains top-k users for item i. Considering the difference types of users that we have defined above, we define three sub-tasks for RNPR:

Expl-RNPR: To find the top-k explore users who are most likely to purchase the given item i, that is, $U_i^t = U_i^{expl}$.

Rep-RNPR: To find top-k repeat users who are most likely to repurchase the given item i, that is, $U_i^t = U_i^{rep}$.

Mixed-RNPR: To find the top-k users who are most likely to purchase the given item i, that is, the target users are simply the set of all users: $U_i^t = U = U_i^{expl} \cup U_i^{rep}$.

3.2 Candidate filtering

Given a specific item i and its target users U_i^t for the RNPR task, the goal of *candidate filtering* is to select a subset of candidate users $\hat{U}_i^t \subseteq U_i^t$ based on their historical interactions S^h. More formally, we seek to define a candidate filtering function g such that

$$\hat{U}_i^t = \{u_1^c, u_2^c, \ldots, u_q^c\} = g(i, S_1^h, S_2^h, \ldots, S_m^h). \tag{2}$$

Given a filtered set of candidate users $\hat{U}_i^t$ for item i, we only compute item scores for users in this filtered set of users $\hat{U}_i^t$ instead of all candidate users U_i^t.

4 REPETITION ANALYSIS

People often have regular habits and display repetition behavior in grocery shopping [3, 25, 26, 42]. Li et al. [25] analyze the repetition behavior from the user side on the item level. That is, how many of the items that the user will purchase next are repeat items that they have purchased before. However, repetition behavior at the category level and from the item side remain unknown. In particular, (i) at the *category level*, how many of the categories from which the user will purchase an item are categories that they have previously purchased an item from? And (ii) from the *item side*, among the users who will purchase the given item or from the given category, what is the proportion of users who have already purchased the given item or from the given category in their previous interactions?

To better understand users' repetition behavior in grocery shopping, we analyze both the item and category level repetition behavior from both the item side and the user side.[4] Specifically, we analyze four types of repeat ratio *RepR*, i.e, user-side item-level $RepR_u^{item}$, user-side category-level $RepR_u^{cat}$, item-side item-level $RepR_i^{item}$ and item-side category-level $RepR_i^{cat}$, defined as follows:

$$RepR_u^{item} = \frac{1}{N}\sum_{n=1}^{N} \frac{\#\mathit{repeat\ items}\ i \in I_{u_n}^h \text{ the user } u_n \text{ will purchase}}{\#\text{all items the user } u_n \text{ will purchase}} \tag{3}$$

$$RepR_u^{cat} = \frac{1}{N}\sum_{n=1}^{N} \frac{\#\mathit{repeat\ categories}\ c \in C_{u_n}^h \text{ the user } u_n \text{ will purchase items from}}{\#\text{all categories the user } u_n \text{ will purchase items from}} \tag{4}$$

$$RepR_i^{item} = \frac{1}{M}\sum_{m=1}^{M} \frac{\#\mathit{repeat\ users}\ u \in U_i^{rep} \text{ who will purchase the given item } i_m}{\#\text{all users who will purchase the given item } i_m} \tag{5}$$

$$RepR_i^{cat} = \frac{1}{Q}\sum_{q=1}^{Q} \frac{\#\mathit{repeat\ users}\ u \in U_c^{rep} \text{ who will purchase items from the given category } c_q}{\#\text{all users who will purchase items from the given category } c_q}, \tag{6}$$

where N, M, and Q are the number of users, items and categories, respectively. We compute these four ratios for each of the two datasets that we will be using in this paper, Instacart and Dunnhumby (see Section 7.2). See Table 2.

From the user side (the first row in Table 2), we can observe that both the item level repeat ratio $RepR_u^{item}$ and category level repeat ratio $RepR_u^{cat}$ are high, ranging from 0.4264 to 0.8791. The results indicate that a large proportion of items/categories the users will purchase in the next period is made up from items/categories that

[4]We perform this analysis by splitting the data into historical baskets and future baskets, which is the same as the experimental setting in Section 7.2

Table 2. Repeat ratios at the item level and category level, from the item perspective and the user perspective.

Dataset	Instacart		Dunnhumby	
Perspective	Item level	Category level	Item level	Category level
User side	0.6822	0.8791	0.4264	0.8737
Item side	0.6111	0.7751	0.4374	0.6649

the users have purchased before. The category level repeat ratio $RepR_u^{cat}$ is relatively high, which shows that the repetition behavior at the category level is more stable than item level in grocery shopping. For example, a user might like to buy fruits every time, but the user might alternate between different types of fruits as time passes.

From the item side, we can also see that both datasets have considerable repeat ratios, i.e., $RepR_i^{item}$ and $RepR_i^{cat}$, ranging from 0.4374 to 0.7751. Similarly, the category level repeat ratio $RepR_i^{cat}$ is also higher than the item level repeat ratio $RepR_i^{item}$. The results indicate that a considerable proportion of the users in our datasets are *repeat users*.

The results presented above indicate that in grocery shopping people have regular habits and that repetition behavior is a strong signal that can be used to address the item-centered RNPR problem from several angles: (i) with the habit module to model users' category level exploration behavior in HIF model (Section 1.1.2); (ii) with the REUR algorithm, which decouples the repetition task and exploration task (Section 5.3); and (iii) with the repetition-rule based candidate filtering methods (Section 6.1) and the CFM model to model category level repetition behavior (Section 6.2) for reducing candidate users.

5 REVERSE NEXT-PERIOD RECOMMENDATION

In this section, we introduce the habit-interest fusion (HIF) model and corresponding training strategies for the Expl-RNPR task, describe a simple time-aware frequency model for the Rep-RNPR task, and finally describe the REUR algorithm for the Mixed-RNPR task.

5.1 Habit-interest fusion model for Expl-RNPR

The objective of a user-centered recommendation model is to rank positive items higher than the negative items, i.e., giving a higher prediction score to positive items, where the prediction score for an item may not represent the user's absolute preference on this item, as this prediction score can be influenced by other items in the catalog and the item distribution in the dataset, e.g., popularity. User-centered recommendation models usually only take a user's historical interactions and learn the general interest of the user and but may not track the users' interest w.r.t. a specific given item as time goes by. To achieve accurate item-centered recommendations, there are two things that should be taken into consideration: (i) the prediction of the model should be appropriate and meaningful for ranking users for a given item; and (ii) apart from a user's historical interactions, the recommendation model should also take the given item as input and be able to track user's interests or habits w.r.t. the given item as time goes by.

5.1.1 Model. Recall from Section 3.1 that the Expl-RNPR task is to find the top-k explore users who are most likely to purchase a given item i. To address the Expl-RNPR task, we propose a habit-interest fusion (HIF) model, which leverages frequency information to model category-level repetition and exploration habits, and pre-trained item representations to model user's interests. Figure 1 illustrates the architecture of the HIF model.

Pre-trained embedding. In the context of NLP, the skip-gram framework [29, 30] has been proposed to learn word representations via predicting the surrounding words within the context. Several recent publications [5, 12, 42] leverage skip-gram techniques to learn item/product representations in an e-commerce scenario. In this paper,

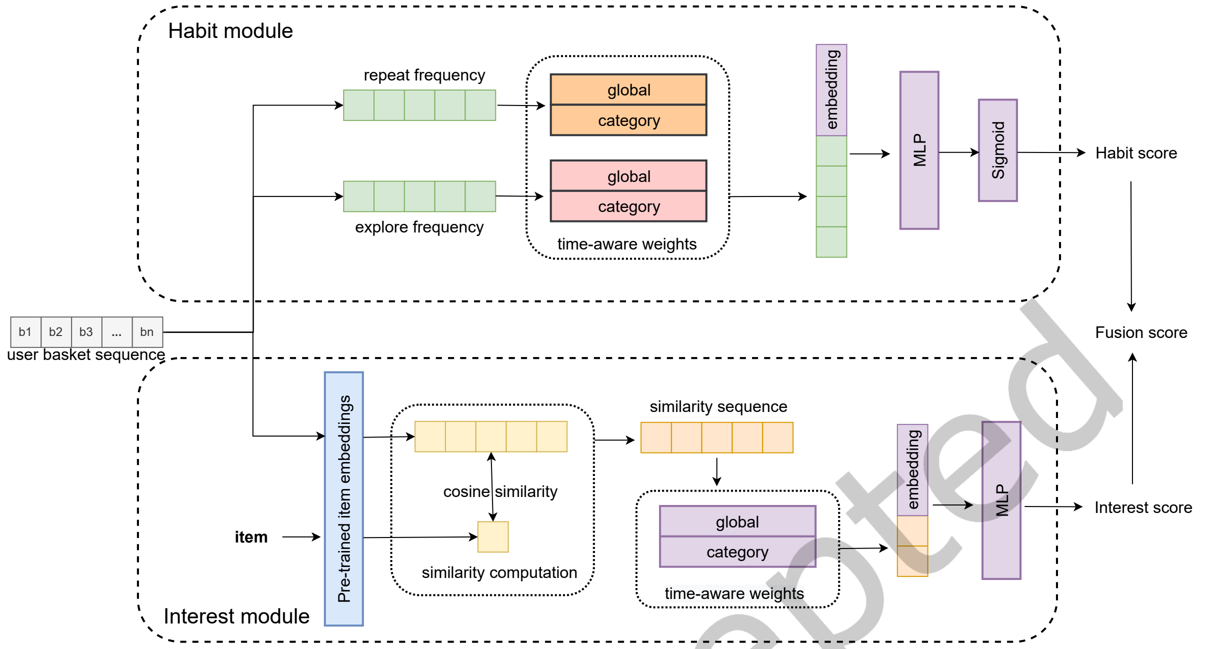


Fig. 1. Overall architecture of the HIF model.

we assume that the items within the same basket share similar semantics and use basket-level skip-grams to derive the embeddings of items. We regard a particular item as a target item $i \in I$ and regard the other items in the same basket as context items $i' \in I_b^i$. Then, the learning objective is to maximize the following function:

$$\mathcal{L} = \sum_{i \in I} \sum_{i' \in I_b^i} \log p(i' \mid i), \tag{7}$$

where $p(i' \mid i)$ denotes the probability of observing a context item $i' \in I_b^v$ given the current/target item i. It is defined by a softmax function:

$$p(i' \mid i) = \frac{\exp(Emb_i^T \cdot Emb_{i'})}{\sum_{m=1}^{M} \exp(Emb_i^T \cdot Emb_{i'_m})}, \tag{8}$$

where Emb_i and $Emb_{i'}$ are vector representations of the current item i and the context item i', respectively. M represents the number of items in the item catalog. After pre-training on historical data, we can get a vector representation (a.k.a. embedding) of each item.

Interest module. Suppose that a user u has a sequence of historical baskets $S^h = \{B^1, B^2, \ldots, B^t\}$. We first get pre-trained item embeddings Emb_i for each item i within each basket B^t. Note that baskets may have different sizes, so we aggregate item embeddings within the same basket by a pooling strategy (max pooling or average pooling) to generate the basket representation Emb_b^t at each timestamp t. Given the target item i we want to recommend, we compute the cosine similarity $Sim_{u,i}^t$ between its embedding Emb_i and basket embedding Emb_b^t at each timestamp, and then get the similarity vector $Sim_{u,i}$, which reflects user's interests in the given item i across different timestamps. That is:

$$Emb_b^t = \text{Pooling}\left(Emb_{i_1^t}, Emb_{i_2^t}, \ldots, Emb_{i_n^t}\right) \tag{9}$$

$$Sim^t_{u,i} = \cos\left(Emb_i, Emb^t_b\right) = \frac{Emb_i \cdot Emb^t_b}{|Emb_i||Emb^t_b|} \tag{10}$$

$$Sim_{u,i} = \left[Sim^1_{u,i}, Sim^2_{u,i}, \ldots, Sim^t_{u,i}\right]. \tag{11}$$

To model users' dynamic interests, we introduce two types of time-aware weight embeddings, i.e., (i) a category specific time-aware weight embedding TW^c_e, which can only be trained by the samples of the corresponding category c, and (ii) a global time-aware weight embedding TW^g_e, which is shared across categories and can be trained by all training samples.[5] For a given item i and user $u \in U^{expl}_i$, we compute the dot products of the similarity vector $Sim_{u,i}$ and two time-aware weight embeddings, i.e., TW^c_e and TW^g_e, to get time-aware interests features, i.e., $SimF^c_{u,i}$ and $SimF^g_{u,i}$. Finally, we concat $SimF^c_{u,i}$ and $SimF^g_{u,i}$ with a trainable category embedding Emb^{inte}_c to get a hybrid representation, which will be fed into a two layer fully-connected network to get the final interests score $Score^{inte}_{u,i}$, that is:

$$SimF^c_{u,i} = TW^c_e \cdot Sim_{u,i} \tag{12}$$

$$SimF^g_{u,i} = TW^g_e \cdot Sim_{u,i} \tag{13}$$

$$Score^{inte}_{u,i} = \mathrm{FFN}(SimF^c_{u,i} \oplus SimF^g_{u,i} \oplus Emb^{inte}_c). \tag{14}$$

Habit module. In the Expl-RNPR task, we aim to find possible *explore users* for a given item. However, there are no direct interactions between the given item i and *explore users* U^{expl}_i, so we cannot directly model *explore users*' habits w.r.t. the item i. In the grocery shopping scenario, every item belongs to a category, and a category can contain many items. We notice that if an item i will be purchased by the user $u \in U^{expl}_i$, it indicates that the user u will purchase and explore the items in category c^i in the next period. Therefore, we aim to model users' repetition and exploration habits w.r.t. the target category c^i of the given item i.

The users' repetition habits within a category can be dynamic across time. Besides, the purchase frequency within a category can also indicate demands of the user. Specifically, to capture the user's repetition habits, we create a category-level repetition frequency vector *RepVec* for category $c^i \in C$ for the user u by considering both temporal information and frequency information. That is,

$$RepVec_{u,c^i} = \left[\sqrt{|I^{c^i} \cap B^1|}, \sqrt{|I^{c^i} \cap B^2|}, \ldots, \sqrt{|I^{c^i} \cap B^t|}\right], \tag{15}$$

where I^{c^i} is the item set within category c^i; B^t is a set of items (basket) that user u purchased at timestamp t. Note that the square root operation is applied to address the problem of varying sizes of baskets in recommendation systems. By taking the square root, the impact of baskets that are too large is reduced, leading to more equitable and balanced frequency information. Then, we derive time-aware category repetition feature $RepF^c_{u,c^i}$ and global repetition feature $RepF^g_{u,c^i}$ as follows:

$$RepF^c_{u,c^i} = TW^c_{rep} \cdot RepVec_{u,c^i} \tag{16}$$

$$RepF^g_{u,c^i} = TW^g_{rep} \cdot RepVec_{u,c^i}, \tag{17}$$

where TW^c_{rep} and TW^g_{rep} are a category time-aware weight embedding and a global time-aware weight embedding, respectively, for modeling repetition behavior.

Note that the user might be loyal to a specific item [42] and uninterested in exploring new items within the same category, e.g., someone might only purchase a specific brand of milk. To model a user's exploration habits

[5]Note that, the time-aware weight embeddings through out paper will be first initialized using a uniform distribution and then updated during the training process.

within a category, we also create an exploration frequency vector $ExplVec_{u,c^i}$ considering the temporal orders, that is:

$$ExplVec_{u,c^i} = \left[\sqrt{|I^{c^i} \cap B^1_{expl}|}, \sqrt{|I^{c^i} \cap B^2_{expl}|}, \ldots, \sqrt{|I^{c^i} \cap B^t_{expl}|}\right], \tag{18}$$

where B^t_{expl} is a set of *explore items* (new items) that the user u purchased at timestamp t.

Similarly, we compute the category exploration feature $ExplF^c_{u,c^i}$ and global exploration feature $ExplF^g_{u,c^i}$ as follows:

$$ExplF^c_{u,c^i} = TW^c_{expl} \cdot ExplVec_{u,c^i} \tag{19}$$

$$ExplF^g_{u,c^i} = TW^g_{expl} \cdot ExplVec_{u,c^i}, \tag{20}$$

where TW^c_{expl} and TW^g_{expl} are the category time-aware weight embedding and the global time-aware weight embedding, respectively, for modeling exploration behavior. Finally, we concatenate repetition features, i.e., $RepF^c_{u,c^i}$ and $RepF^g_{u,c^i}$, exploration features, i.e., $ExplF^c_{u,c^i}$ and $ExplF^g_{u,c^i}$, and a trainable category specific embedding Emb^{hab}_c to get a feature vector, which will be fed into a two-layer fully-connected network to get the habit score $Score^{hab}_{u,i}$. That is,

$$Score^{hab}_{u,i} = \text{FFN}(RepF^c_{u,c^i} \oplus RepF^g_{u,c^i} \oplus ExplF^c_{u,c^i} \oplus ExplF^g_{u,c^i} \oplus Emb^{hab}_c) \tag{21}$$

Fusion. We compute the fusion score by:

$$Score^{fusion}_{u,i} = \text{Sigmoid}(Score^{hab}_{u,i}) \cdot Score^{inte}_{u,i}. \tag{22}$$

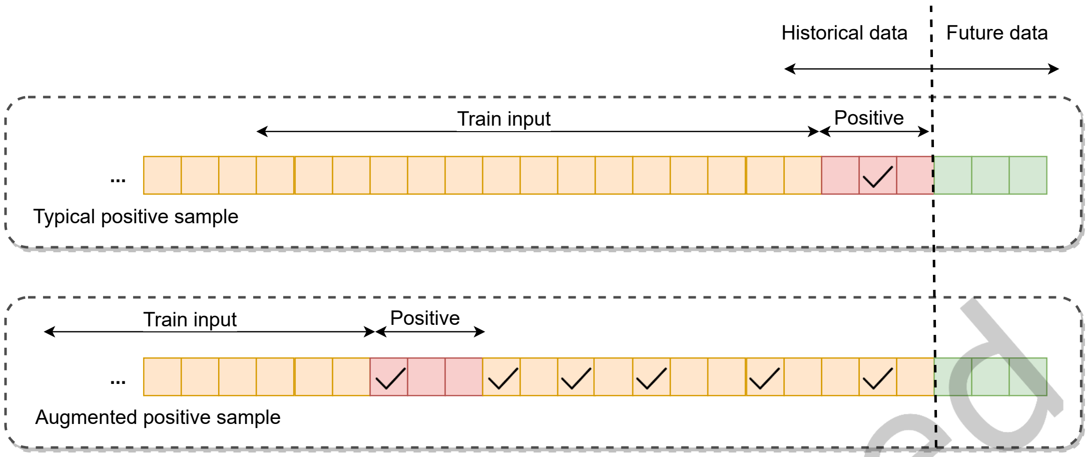
5.1.2 *Training.* In a conventional *user-centered* scenario, a recommendation model is optimized based on a user-wise loss, which is computed based on all items for each user. Since we focus on *item-centered* recommendations to rank users for the given item, we propose an item-wise ranking loss to train our model. Specifically, positive users and negative users are sampled for each item, and then the training objective is to minimize the following loss function:

$$\mathcal{L}_i = -\frac{1}{N}\sum_{k=1}^{N} \log\left(\frac{1}{1+e^{-(Score^k_{pos,i} - Score^k_{neg,i})}}\right), \tag{23}$$

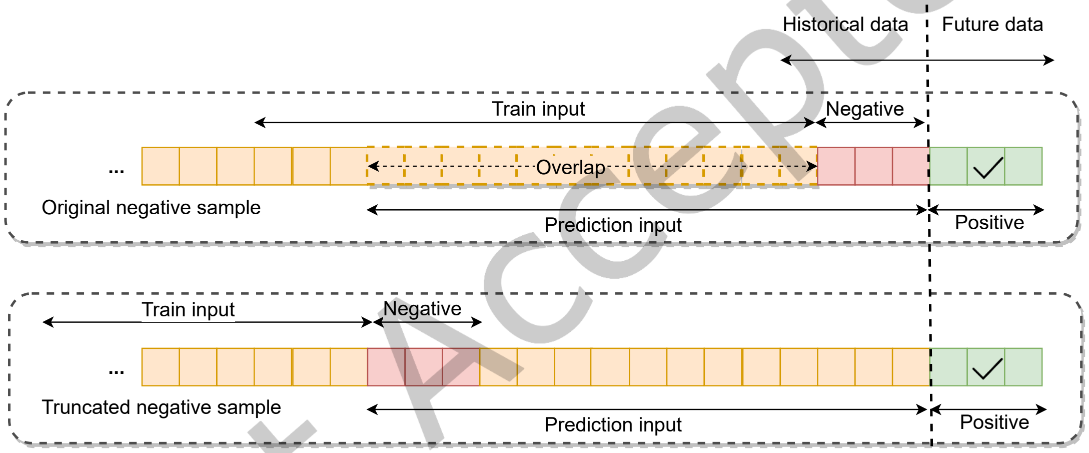
where $Score_{pos,i}$ and $Score_{neg,i}$ represent the predicted fusion scores for positive users and negative users, respectively. By minimizing the proposed item-wise ranking loss, the model will maximize the difference in predicted preference (fusion) scores between the positive and negative users, such that positive users are ranked higher in the predicted user ranking list. Even though the definition of item-wise ranking loss is straight forward, we identify two major issues w.r.t. the training process of the Expl-RNPR model using item-wise ranking loss.

First, as illustrated in Figure 2a, a typical positive sample for Expl-RNPR is an *explore user* who only purchased the given item i in the last period of the historical sequence. However, as shown in Table 3, items have a small number of such positive samples (i.e., new users) if we only consider the last period of the historical sequence. Therefore, we need to augment the positive samples, i.e., include more users who explore the target item for the first time. According to an intuitive reading of the Expl-RNPR task, we should not select a *repeat user* who has already purchased the given item as a positive training sample for the given item, since Expl-RNPR is targeting *explore users*. However, *repeat users* of the target item should have a sub-sequence of interactions, i.e., a basket sequence before their first purchase, that could be regarded as a positive sample for Expl-RNPR training (shown in Figure 2a).

Second, a typical negative sample for Expl-RNPR is an *explore user* who did not purchase the given item i in the last period sequence. However, as illustrated in Figure 2b, if a user u is a new user of a given item i, i.e., $i \in I^n_u$, the user has not purchased this item in previous interactions, i.e., $i \notin I^h_u$, and this means that the user



(a) Positive sample augmentation strategy.



(b) Negative sample adjustment strategy.

Fig. 2. Training samples construction strategies.

should also be regarded as a negative sample for the item *i* during the training process. In this case, when we use a leave-one/few-out splitting strategy to construct a historical (training) dataset and a future (test) dataset, the positive samples (i.e., the ground-truth) in the test set will be the negative sample in the training set, even though they may share a long overlap between two input sequences. To avoid the negative impact of this case, we propose a negative sample adjustment strategy, which eliminates the potential overlap between positive and negative sequences by truncating a sub-sequence from the original negative samples. Note that we perform the truncation action on all negative samples, since we do not know which one is the positive sample in the future (test) dataset.

5.2 Time-aware frequency model for Rep-RNPR

The task of Rep-RNPR is to help a given item find *repeat users*. Different from *explore users*, *repeat users* have previously had direct interactions with the item that we want to recommend. Therefore, we employ a simple

Table 3. The number of training samples for Expl-RNPR.

Dataset	Avg. #negative samples per item	Avg. #positive samples per item	Avg. #positive samples per item after positive augmentation
Instacart	1,307.9	5.8	31.9
Dunnhumby	667.9	6.0	73.6

time-aware frequency model, which only uses users' direct interactions with the given item instead of using item-item correlations and complex representations. Formally,

$$Score_{u,i} = \sum_{j=1}^{m} F_{u,i,j} \cdot \beta^{T-j}, \tag{24}$$

where $F_{u,i,j}$ denotes the user's u purchase frequency of item i at timestamp j, β denotes the time-decay factor, which emphasizes the impact of recent interactions. We find the optimal β based on the historical data.

5.3 The REUR algorithm for Mixed-RNPR

Different from the Expl-RNPR task in Section 5.1 and the Rep-RNPR task in Section 5.2, the Mixed-RNPR task considers both *repeat users* and *explore users* for the items that are to be recommended. Theoretically, a model for Expl-RNPR can also be applied to Mixed-RNPR without excluding *repeat users* in the final prediction stage. A recent analysis [25] shows that the repetition and exploration tasks in the (user-centered) NBR problem have different levels of difficulty, where the repetition task, i.e., recommending repeat items to a user, is a much easier task. In an item-centered recommendation scenario, we mainly use item-to-item relations to infer *explore users*' interests for the target item, since *explore users* do not have any previous interactions with it. Yet, we can address *repeat users* prediction via the users' direct interactions with the target item.

Considering the above differences between the repetition and exploration tasks in Mixed-RNPR, we propose a *repetition-exploration user ranking* (REUR) algorithm to decouple the two tasks and investigate the trade-off between repetition and exploration in an item-centered setting. Specifically, as in shown in Algorithm 1, we use separate models for the repetition and exploration tasks.[6] Note that the models designed in Section 5.1 and Section 5.2 can be used for ranking *explore users* and ranking *repeat users*, respectively. For a given item, we rank *repeat users* and *explore users* according to the scores derived from the repetition model M^{rep} and exploration model M^{expl}, respectively. Then, REUR generates the final ranked list of users P_i^n by combining the above two ranked lists. We define a combination (repeat) ratio α, which controls the proportions of *repeat users* and *explore users*. Assume that we want to recommend a given item to k users. Then REUR first selects the top-m highest-score *repeat users* and then fills any remaining slots with the top-n highest-scoring *explore users*, where $m = k \cdot \alpha$ and $n = k - m$.

As we will see, one simple way to achieve good performance on the Mixed-RNPR task is to find a global optimal combination ratio α for all items. We notice that different items might have different repurchase tendencies, i.e., the repurchase tendency of a pan is likely to be smaller than that of the milk that is cooked in it, which might influence the optimal combination ratio. The repurchase tendency is defined by:

$$RT_i = \frac{\text{\#users who repurchase item } i}{\text{\#users who bought item } i \text{ before}}. \tag{25}$$

[6]For a given item, the repetition task is to find repeat users, whereas the exploration task is to find new users.

```
Algorithm 1: Repetition-exploration user ranking algorithm
  Data: User set $U$, item set $I$, basket sequences $S$, user list size $k$, combination/repeat ratio $\alpha$
  Result: Predicted top-$k$ users $P_i^n$ for item $i$.
1 Get repetition model $M^{rep}$ by finding optimal time-decay factor $\beta$ on the dataset;
2 Get exploration model $M^{expl}$ via training the HIF model on the dataset until converge;
3 for each given item $i$ do
4     Get repeat users $U_i^{rep}$ and explore users $U_i^{expl}$;
5     Rank repeat users $u \in U_i^{rep}$ via $M^{rep}(S_u, i)$;
6     Rank explore users $u \in U_i^{expl}$ via $M^{expl}(S_u, i)$;
7     Decide number of repeat users, i.e., $m = k \cdot \alpha$, and explore users, i.e, $n = k - m$;
8     Construct the top-$k$ users $P_i^n$ using top-$m$ repeat users and top-$n$ explore users;
9 end
```

We also cluster items according to their repurchase tendency $RT_i \in [0, 0.2), [0.2, 0.4), [0.4, 0.6), [0.6, 0.8), [0.8, 1.0]$, and try to find the optimal combination ratio α for each cluster. We sweep the combination ratio α and select the optimal α based on the performance of the validation set.

Apart from achieving good performance on the Mixed-RNPR task, another important task is to investigate the trade-off between repetition and exploration so as to make sure that we gain an in-depth understanding of the potential imbalances in the RNPR task. With the REUR algorithm, we can also easily investigate the trade-off by setting different combination ratios (see Section 8.4).

6 CANDIDATE FILTERING

Given a specific item i, an intuitive solution to an item-centered recommendation task is to compute scores for every user in the target user set U_i^t. However, this is usually computationally expensive. The goal of candidate filtering is to select a group of users as candidates, which is a subset of target users, i.e., $\hat{U}_i \in U_i^t$. After candidate filtering, we can reduce the computational costs by only considering users within the candidate set.

6.1 Repetition-rule based candidate filtering

According to the repetition analysis in Section 4, users have regular habits in grocery shopping and category-level repetition behavior seems more stable and prominent than item-level repetition behavior. Next, we first propose two repetition rule-based candidate filtering methods, i.e., RRBF-item and RRBF-cat. Formally,

RRBF-item: For a given item i, we only select the *repeat users* U_i^{rep} to form up the candidate set, that is, $\hat{U}_i = U_i^{rep}$. This method is designed for Mixed-RNPR, which helps to reduce Mixed-RNPR to Rep-RNPR.[7]

RRBF-cat: For a given item i, we get its corresponding category c^i and only select the explore users U_i^{expl} who have previously purchased items from c^i to form the candidate set, that is, $\hat{U}_i = U_c^{rep} \cap U_i^{expl}$. Note that RRBF-cat is used in the Expl-RNPR task.

6.2 Model-based candidate filtering

Repetition-rule based filtering (RRBF) does not consider the temporal information and is static. To further reduce the number of candidates on top of RRBF-cat, we propose a *candidate filtering model* (CFM) to predict whether a user likes to repurchase the category or not, then select the users who would like to purchase the category of the

[7]RRBF-item cannot be used for Expl-RNPR, since it only selects repeat users as candidates.

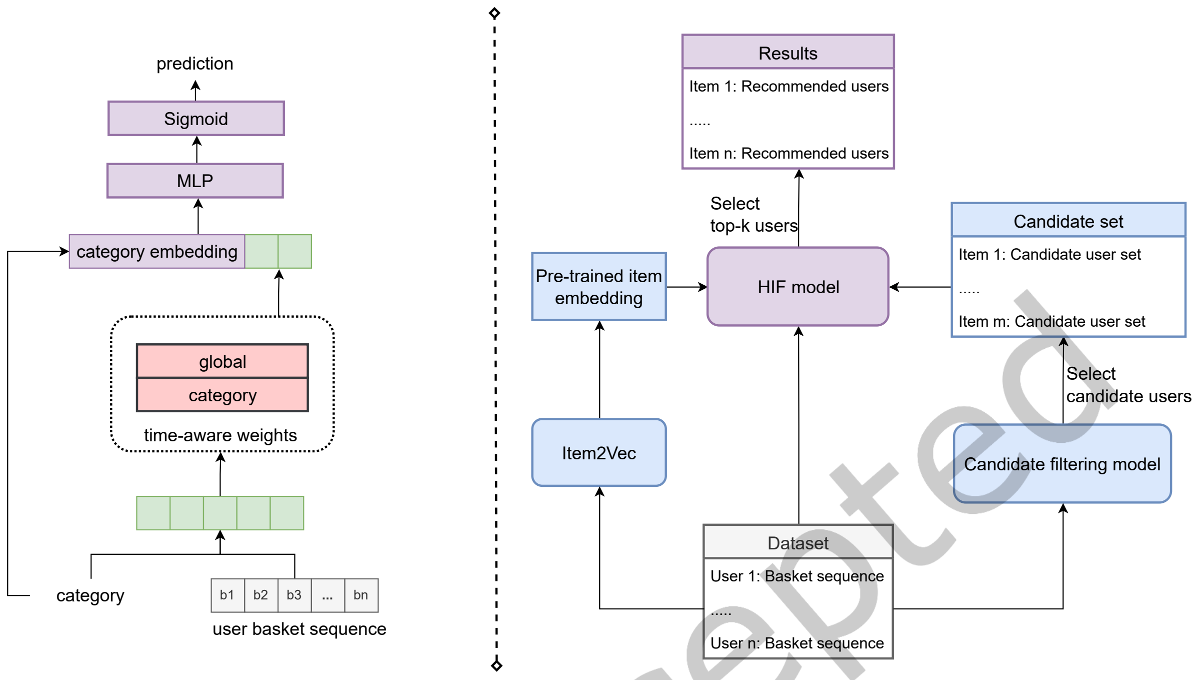


Fig. 3. The architecture of candidate filtering model (left) and the overall pipeline (right) of Expl-RNPR task.

given item. The architecture of the candidate filtering model is shown in Figure 3. Note that the category catalog is relatively stable and small compared to the item catalog, and the items within the same category can share the same set of candidate users.

CFM predicts users' repurchase behavior by modeling their dynamic demands within the target category. For the sake of simplicity, we do not consider dependencies among different categories. Specifically, we use the category-level repetition frequency vector $RepVec$ of category $c \in C$ (defined in Eq. 15), which contains both temporal information and frequency information.

Different categories might have different characteristics w.r.t. repurchase behavior. For example, daily necessities like fruit might be purchased during every visit to the grocery store, whereas users are less likely to repurchase household items like dish soap right after their previous purchase of dish soap. Therefore, we introduce a category specific time-aware weight embedding TW^{c}_{cf} to model temporal dependencies. In addition, we also introduce a global time-aware weight embedding TW^{g}_{cf} which can be shared and trained by all categories, so that different categories can benefit from the training samples of each other. Given a use u and a category c, we first derive the category specific repetition feature $RepF^{c}_{u,c}$ and the general repetition feature $RepF^{g}_{u,c}$, then we concat $RepF^{c}_{u,c}$ and $RepF^{g}_{u,c}$ with a category embedding Emb^{c}_{cf}, and feed it to a two layer fully-connected neural network, that is:

$$RepF^{c}_{u,c} = TW^{c}_{cf} \cdot RepVec_{u,c} \tag{26}$$

$$RepF^{g}_{u,c} = TW^{g}_{cf} \cdot RepVec_{u,c} \tag{27}$$

$$p_{u,c} = \text{Sigmoid}(\text{FFN}(RepF^{c}_{u,c} \oplus RepF^{g}_{u,c} \oplus Emb^{c}_{cf})), \tag{28}$$

where $p_{u,c}$ is the probability that user u will purchase items within the category c; $\oplus$ denotes the concatenation operation.

Once we obtain the repurchase probabilities $p_{u,c}$ of the target users, we can set a filtering threshold λ to filter candidates. For a given item $i \in I^c$, we only select users whose $p_{u,c}$ is above the filtering threshold λ as candidates. The overall pipeline of Expl-RNPR is shown in Figure 3.

6.2.1 Training. In the training set, every user has a ground-truth label for each category, i.e., whether the user has repurchased from the category or not. Conversely, for a specific category, we can split users into positive users and negative users. However, positive users and negative users within a category might be imbalanced. Besides, the positive users are unevenly distributed across all categories, e.g., a popular category is likely to have more positive users than a less popular category.

To overcome the problems of imbalanced data listed above, we sample the same number of training instances (users) U^c for each category in every epoch instead of using all users, and balance the number of positive users and negative users within each category. Then, we use binary cross-entropy loss to train our model:

$$\mathcal{L}_c = -\frac{1}{|U^c|} \sum_{u \in U^c} y_{u,c} \log(p_{u,c}) + (1 - y_{u,c}) \log(1 - p_{u,c}), \tag{29}$$

where $y_{u,c}$ and $p_{u,c}$ denote the ground-truth of category c and the probability of c being purchased by user u in the next period.

7 EXPERIMENTAL SETUP

In this section, we describe our experimental settings, including our research questions, datasets, constructed baselines, parameter settings, and evaluation metrics.

7.1 Research questions

To better understand the RNPR problem and investigate the effectiveness of the proposed methods, we intend to answer the following questions through our experiments:

(RQ1) How do user-centered state-of-the-art NBR methods perform on the RNPR task?
(RQ2) What is the effectiveness of our newly proposed methods? Do they outperform existing baselines?
(RQ3) What is the effectiveness of our training strategies for the HIF model in Expl-RNPR?
(RQ4) What are the differences and trade-offs between the repetition and exploration tasks in Mixed-RNPR?
(RQ5) Do the proposed candidate filtering strategies help to reduce computational costs at inference time? How does the candidate filtering process influence the performance of our models?

7.2 Datasets

In order to ensure the reproducibility of our study, we conduct our experiments on two publicly available real-world datasets:

Dunnhumby: covers two years of household-level transactions at a retailer from a group of 2,500 households. All products bought by the same customer in the same transaction are treated as a basket.[8]

Instacart: contains over three million grocery orders of Instacart users. We treat all items purchased in the same order as a basket.[9]

In each dataset, we sample active users with at least 30 baskets in the dataset, and truncate a basket sequence to 100 baskets. We follow a strategy that is similar to the widely used leave-one-out approach to split the dataset.

[8]https://www.dunnhumby.com/source-files/
[9]https://www.kaggle.com/c/instacart-market-basket-analysis/data

Table 4. Dataset statistics after preprocessing.

Dataset	Users	Categories	Target items	Avg. #items per basket	Avg. #baskets per user	Avg. #item per user	Avg. #target users per target item
Instacart	30,134	134	1,369	10.19	49.47	142.38	442.47
Dunnhumby	1,991	307	866	11.77	77.44	528.53	39.27

Specifically, for each user, the last 10 baskets are regarded as future baskets, and all remaining baskets are regarded as historical baskets. All training is conducted using the historical data. We select target items according to their frequency in the ground-truth (a.k.a. future data) to ensure there are new users for this item in the next period, otherwise we can only add zero to the evaluation metrics. Note that using the explore users' frequency in the ground-truth instead of the repeat users' not only allows us to address the Expl-RNPR task, but also to make a fair comparison between the Expl-RNPR task and the Mixed-RNPR task. Considering the number of users in each dataset, the minimum number of future new users' for an item is set to 50 for the Instacart dataset, and 10 for the Dunnhumby dataset. Since we use category information in our model, splitting across items has a risk of information leakage. So we split the items into validation and test dataset according to their corresponding category [11], 50% categories for validation, and 50% categories for testing. We repeat the split 5 times and report the average performance w.r.t. metrics over the 5 splits.

7.3 Methods used for comparison

We construct three simple methods, i.e., Random, I-TopFreq and C-TopFreq, two pre-training based methods, i.e., Basket2Vec and User2Vec, and select three SOTA NBR methods according to [25] and [2], i.e., DNNTSP [54], TIFUKNN [17] and ReCANet [2], as baseline methods for comparison:

7.3.1 Simple methods. We select three simple baseline models:

- Random selects the users for the given item at random.
- I-TopFreq ranks users according to their historical purchase frequency of the given input item, and selects users with the top-k highest purchase frequency. Since this method can only select *repeat users*, it will not be used in the Expl-RNPR task.[10]
- C-TopFreq ranks users according to their historical purchase frequency of different items within the given item's category, and selects users who prefer to purchase a lot of different items within the input item's category. This method is designed for the Expl-RNPR task, and it will not be used in the Rep-RNPR task.

7.3.2 Pre-training based methods. We select two pre-training based methods:

- Basket2Vec [42] pre-trains the item embeddings at the basket level, uses the average embedding of the items in historical baskets to represent the user, computes similarity between the user and the given item, and finally selects the top-k users who have the highest similarity with the given item.
- User2Vec [42] pre-trains the item embeddings on the user level, uses the average embedding of the items in historical baskets to represent the user, computes similarity between the user and the given item, and finally selects the top-k users who have the highest similarity with the given item.

7.3.3 Next-basket recommendation methods. We select three NBR methods as baselines:

[10]Repeat users play a vital role in Rep-RNPR and Mixed-RNPR, and item frequency serves as a reliable signal for identifying them for a particular item. We evaluated a similar category frequency-based method, but found that it did not outperform I-TopFreq w.r.t. Rep-RNPR and Mixed-RNPR, since using category frequency could result in noise and dilute the significance of item frequency information.

- DNNTSP [54] is a state-of-the-art NBR method, which encodes the item-item relation via a GNN, and models temporal dependencies via self-attention techniques. After training DNNTSP, we first derive the target users' purchase probability of the items, and then select the top-k users with the highest purchase probability of the given item.[11]
- TIFUKNN [17] is a state-of-the-art NBR method, which constructs personal item frequency information (PIF) for each user, and leverages a KNN-based method based on PIF. We first get target users' scores of the item, then select the top-k users with the highest purchase score of the given item.
- ReCANet [2] is a state-of-the-art NBR method, which builds a neural-based temporal model to focus on recommending repeat items to the user in the NBR task. After training ReCANet, we first derive the target users' purchase probabilities of items, and then select the top-k users with the highest purchase probability of the given item.

7.4 Evaluation metrics

To assess the performance of RNPR methods, we extend three widely used user-centered metrics to the item-centered setting and arrive at the following metrics: *Recall@K*, *nDCG@K* and *IHN@K*.

Recall measures the ability to find all users who will purchase the given item in the next period:

$$Recall@K = \frac{1}{N}\sum_{j=1}^{N}\frac{\left|P_{i_j} \cap T_{i_j}\right|}{\left|T_{i_j}\right|}, \tag{30}$$

where P_{i_j} are the top-K predicted users for item i_j, and T_{i_j} denotes ground-truth users for item i_j.

nDCG is a ranking metric that also considers the order of the users:

$$nDCG@K = \frac{1}{N}\sum_{j=1}^{N}\frac{\sum_{k=1}^{K} p_k/\log_2(k+1)}{\sum_{k=1}^{\min(K,|T_{i_j}|)} 1/\log_2(k+1)}, \tag{31}$$

where p_k equals 1 if $P_{i_j}^k \in T_{i_j}$, otherwise $p_k = 0$. $P_{i_j}^k$ denotes the k-th user in the predicted user list P_{i_j} for item i_j.

IHN represents the average number of correct users the model can find for each item, that is:

$$IHN@K = \frac{1}{N}\sum_{j=1}^{N}\left|P_{i_j} \cap T_{i_j}\right|. \tag{32}$$

Since the two datasets that we use, Instacart and Dunnhumby, have different numbers of users, when we evaluate the Expl-RNPR and Mixed-RNPR performance, the value of K for the Instacart dataset is set to 100 and 200, the value of K for the Dunnhumby dataset is set to 50 and 100. Since the number of repeat user candidates is limited, the K is set to 20 and 50 for Rep-RNPR task.

7.5 Configuration

For all experiments, we set the next period size to 5 and 10 baskets. For Basket2Vec and User2Vec the embedding size is set to 100 for all datasets. For TIFUKNN, the number of neighbors is selected from [100, 200, 300, 400, 500],

[11]When training DNNTSP, we use the binary cross entropy loss from the original paper [54] rather than devising a new user-centered loss compatible with their approach. There are several reasons for this. First, we want to explore the performance of user-centered model in the item-centered setting. Second, our proposed item-wise loss will only sample some of the users for training, which would be unfair for the DNNTSP model, since the HIF model can use all available users in its pre-training stage. Third, DNNTSP will learn and update item embeddings during the training process, which makes it hard to pre-compute user features to speed up training process and item-wise loss can only use one item's loss information at each backpropagation step, so training DNNTSP model via the proposed item-wise loss is extremely slow.

the fusion weight and time-decay factor are selected from $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$, and the window size is set to the next period size, i.e., 5 or 10.

DNNTSP and ReCANet are used with the same parameter settings as in [54] and [2], respectively. For HIF, we use the same pre-trained item embeddings as Basket2Vec to make a fair comparison with it. For both the HIF model and CFM model, the hidden layer of the fully-connected network is set to 32, the maximum user sequence is set to 30. For REUR, the time-decay factor β is chosen from $[0.5, 0.6, 0.7, 0.8, 0.9]$, we sweep the combination ratio α in $[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$ to investigate the trade-off and find the optimal α for Mixed-RNPR. We use Adam optimizer with 0.001 as learning rate and 256 as batch size to train our models.

We use PyTorch to implement our model and train it using a TITAN X GPU with 12G memory. We repeat our experiments 5 times and report the average results. The code for the HIF model and CFM model is based on Pytorch[12] and we share the code and parameters in a public GitHub repository.[13]

8 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we report on the experimental results to answer the research questions listed in Section 7.1.

8.1 RQ1: Performance comparison of existing "user-centered" methods

Table 5, 6 and 7 show the performance of all approaches for the Expl-RNPR task, Rep-RNPR task and the Mixed-RNPR task, respectively. We have several findings based on the experimental results. As expected, the performance of Random is always among the lowest in terms of all metrics for all tasks, as it just randomly selects users. C-TopFreq outperforms Random on the Expl-RNPR task since it models users' interests w.r.t. the category of the target item. I-TopFreq achieves competitive or even better performance compared to other approaches on both the Rep-RNPR and Mixed-RNPR task, which confirms that item repetition frequency information is important and only considering *repeat users* can achieve quite good performance on the Mixed-RNPR task.

Among the two pre-training based methods, Basket2Vec always achieves better performance than User2Vec in all cases, which indicates that basket-level pre-training can get better item representations than user-level pre-training. We suspect that this is because users' interests are dynamic and items purchased at the same time have more similarity. Basket2Vec is the best performing approach on the Expl-RNPR task, but Basket2Vec is inferior to I-TopFreq on both the Rep-RNPR and Mixed-RNPR task, which suggests that the item-item correlation is less important than the item repetition frequency information in these two tasks.

Surprisingly, the state-of-the-art DNNTSP method performs poorly on the Expl-RNPR task; its performance is in the same range as that of Random. DNNTSP is supposed to capture item-item correlations effectively, as it leverages advanced techniques, i.e., a GNN and self-attention mechanism. The ReCANet method achieves quite good performance on the Rep-RNPR task and the Mixed-RNPR task, which further emphasizes the importance of modeling repeat users in the Mixed-RNPR task. However, ReCANet does not outperform the SimpleTF method w.r.t. these two item-centered tasks. A plausible reason is that the user-wise binary cross entropy loss tries to distinguish items for a given user, which is sub-optimal for "item-centered" recommendations, where the goal is to distinguish users for a given item. Compared to DNNTSP, the relatively simple TIFUKNN is more robust on the Expl-RNPR task and even achieves the best performance (amongst the baselines) on the Dunnhumby dataset and the second best performance on the Instacart dataset. For the Rep-RNPR task and Mixed-RNPR task, TIFUKNN achieves good performance on the Instacart dataset, as it adopts personal item frequency information. However, it performs worse than the simple I-TopFreq on Dunnhumby; we suspect that the underlying reason is that the KNN module has a negative impact on finding repeat users in the Rep-RNPR task and Mixed-RNPR task.

[12]https://pytorch.org/
[13]https://github.com/liming-7/RNPR-Rec

Table 5. Exlp-RNPR results of HIF compared against the baselines. Boldface and underline indicate the best performing model and the best performing baseline, respectively. Significant improvements of HIF over the best performing baseline results are marked with † (paired t-test, $p < 0.05$). ▲% shows the improvements of HIF against the best performing baseline.

Dataset	Period	Metric	Baselines						Ours	
			Random	C-TopFreq	Basket 2Vec	User 2Vec	DNNTSP	TIFUKNN	HIF-max	HIF-mean (▲%)
Instacart	5	Recall@100	0.0036	0.0068	0.0211	0.0119	0.0035	0.0148	0.0161	**0.0251**†(19.0)
		nDCG@100	0.0055	0.0079	0.0276	0.0146	0.0055	0.0232	0.0213	**0.0346**†(25.4)
		IHN@100	0.4909	0.6932	2.3655	1.2823	0.4962	2.0027	1.9109	**2.9315**†(23.9)
		Recall@200	0.0067	0.0135	0.0388	0.0237	0.0070	0.0268	0.0307	**0.0444**†(14.4)
		nDCG@200	0.0067	0.0114	0.0359	0.0207	0.0071	0.0275	0.0281	**0.0427**†(15.9)
		IHN@200	0.9708	1.4125	4.4617	2.5971	1.0163	3.6394	3.6685	**5.2537**†(17.8)
	10	Recall@100	0.0033	0.0096	0.0206	0.0115	0.0039	0.0147	0.0158	**0.0237**†(15.0)
		nDCG@100	0.0092	0.0134	0.0453	0.0237	0.0104	0.0389	0.0412	**0.0566**†(24.9)
		IHN@100	0.9142	1.1718	4.4471	2.4161	1.0094	3.7418	3.9003	**5.3562**†(20.4)
		Recall@200	0.0072	0.0182	0.0376	0.0225	0.0071	0.0266	0.0295	**0.0423**†(12.5)
		nDCG@200	0.0107	0.0173	0.0490	0.0274	0.0109	0.0396	0.0431	**0.0582**†(18.7)
		IHN@200	1.9561	2.2505	8.3443	4.8073	1.9528	6.8362	7.2453	**9.6829**†(16.0)
Dunnhumby	5	Recall@50	0.0292	0.0420	0.0623	0.0286	0.0284	0.0647	0.0756	**0.0841**†(30.0)
		nDCG@50	0.0193	0.0270	0.0410	0.0174	0.0185	0.0442	0.0530	**0.0600**†(35.7)
		IHN@50	0.4690	0.5893	1.0022	0.4506	0.4548	1.0305	1.1617	**1.3001**†(26.2)
		Recall@100	0.0569	0.0786	0.1134	0.0637	0.0560	0.1146	0.1323	**0.1514**†(32.1)
		nDCG@100	0.0304	0.0416	0.0620	0.0320	0.0298	0.0648	0.0763	**0.0877**†(26.1)
		IHN@100	0.9154	1.1224	1.8280	1.0329	0.9055	1.8261	2.0620	**2.3643**†(22.7)
	10	Recall@50	0.0290	0.0423	0.0584	0.0305	0.0278	0.0642	0.0753	**0.0848**†(32.1)
		nDCG@50	0.0247	0.0351	0.0493	0.0244	0.0233	0.0578	0.0657	**0.0773**†(33.7)
		IHN@50	0.8952	1.3482	1.7667	0.9280	0.8506	1.9304	2.1937	**2.5056**†(29.8)
		Recall@100	0.0562	0.0791	0.1128	0.0643	0.0566	0.1168	0.1319	**0.1490**†(27.6)
		nDCG@100	0.0374	0.0569	0.0761	0.0410	0.0369	0.0829	0.0912	**0.1082**†(30.5)
		IHN@100	1.7369	2.681	3.4384	1.9602	1.7557	3.4876	3.9814	**4.4191**†(26.7)

To sum up, the performance of the state-of-the-art NBR methods does not always generalize to the Expl-RNPR task, and they are overly complex for the Rep-RNPR task and the Mixed-RNPR task.

8.2 RQ2: Performance of our proposed methods

For the Expl-RNPR task, the performance of the HIF model is shown in Table 5 and Figure 4a. We can make two main observations based on the results. First, HIF with average pooling strategy (HIF-mean) significantly outperforms all existing approaches on both datasets across all metrics, with improvements ranging from 14.4% to 35.7%, since HIF models users' interests via item-item correlations and models habits via frequency information at the same time.[14] Second, the performance of HIF is different when using different basket pooling strategies, and

[14]All occurrences of HIF, unless otherwise stated, refer to HIF-mean, i.e., the HIF model with average pooling strategy.

Table 6. Rep-RNPR results of the simple time-aware frequency model (SimpleTF) and the baselines. Boldface and underline indicate the best and the second best performing model, respectively. Significant improvements of SimpleTF over the best performing baseline results are marked with † (paired t-test, $p < 0.05$). ▲% shows the improvements of SimpleTF against the best performing baseline.

Dataset	Period	Metric	Random	I-TopFreq	Basket 2Vec	User 2Vec	DNNTSP	TIFUKNN	ReCANet	SimpleTF (▲%)
Instacart	5	Recall@20	0.0262	0.0975	0.0535	0.0408	0.1020	0.1050	0.1092	**0.1129**†(3.4)
		nDCG@20	0.1949	0.6928	0.4528	0.3589	0.7545	0.7598	0.7692	**0.7972**†(3.6)
		IHN@20	3.9027	13.3096	8.5759	6.8230	14.5037	14.6265	14.8567	**15.3445**†(3.2)
		Recall@50	0.0661	0.1987	0.1165	0.0919	0.2071	0.2124	0.2191	**0.2264**†(3.3)
		nDCG@50	0.1983	0.6319	0.4125	0.3296	0.6845	0.6886	0.7024	**0.7249**†(3.2)
		IHN@50	9.7476	29.2964	19.1395	15.3787	31.8664	32.0061	32.5761	**33.6926**†(3.4)
	10	Recall@20	0.0265	0.0806	0.0486	0.0383	0.0808	0.0842	0.0893	**0.0922**†(3.3)
		nDCG@20	0.2714	0.7795	0.5421	0.4441	0.8205	0.8292	0.8458	**0.8617** (1.8)
		IHN@20	5.4067	15.1624	10.4104	8.5298	15.9418	16.1461	16.4628	**16.9079**†(2.7)
		Recall@50	0.0656	0.1715	0.1083	0.0883	0.1718	0.1778	0.1838	**0.1897**†(3.2)
		nDCG@50	0.2707	0.7230	0.5005	0.4136	0.7566	0.7644	0.7841	**0.7998** (2.0)
		IHN@50	13.4393	34.4646	23.7418	19.7032	36.1090	36.4683	37.2163	**38.2577**†(2.8)
Dunnhumby	5	Recall@20	0.0845	0.2674	0.1374	0.1027	0.2205	0.2183	0.2670	**0.2821**†(5.4)
		nDCG@20	0.1072	0.4047	0.1973	0.1394	0.3457	0.3252	0.4051	**0.4283**†(5.7)
		IHN@20	1.9041	6.4474	3.3213	2.3951	5.4764	5.2513	6.4471	**6.8007**†(5.5)
		Recall@50	0.2081	0.4516	0.2944	0.2383	0.3983	0.4116	0.4500	**0.4734**†(4.8)
		nDCG@50	0.1580	0.4481	0.2534	0.1944	0.3896	0.3825	0.4479	**0.4711**†(5.1)
		IHN@50	4.7051	11.5646	7.1853	5.7145	10.1565	10.1866	11.4604	**12.0355**†(4.1)
	10	Recall@20	0.1076	0.2854	0.1618	0.1219	0.2379	0.2436	0.2867	**0.3049**†(6.3)
		nDCG@20	0.1575	0.4767	0.2544	0.1785	0.4017	0.3875	0.4783	**0.5077**†(6.1)
		IHN@20	2.9360	7.8469	4.4430	3.2380	6.6009	6.5305	7.8477	**8.3416**†(6.2)
		Recall@50	0.2647	0.4860	0.3564	0.2932	0.4430	0.4587	0.4856	**0.5096**†(4.8)
		nDCG@50	0.2206	0.5097	0.3197	0.2463	0.4463	0.4442	0.5092	**0.5379**†(5.5)
		IHN@50	7.1424	14.3655	9.7911	7.8676	12.7583	12.9982	14.3637	**14.9963**†(4.3)

the average pooling strategy has a clear advantage over the max pooling strategy on both datasets. This result indicates that the average pooling retains more useful information and leads to better basket representations in the HIF model.

For the Rep-RNPR task, the performance of SimpleTF is shown in Table 6.[15] Surprisingly, the SimpleTF method outperforms many complex neural/representation-based methods, with improvements ranging from 1.8% to 6.3%. This result indicates that not all user-centered techniques, e.g., learning item representations and leveraging neighbor information, are helpful in the item-centered recommendation setting.

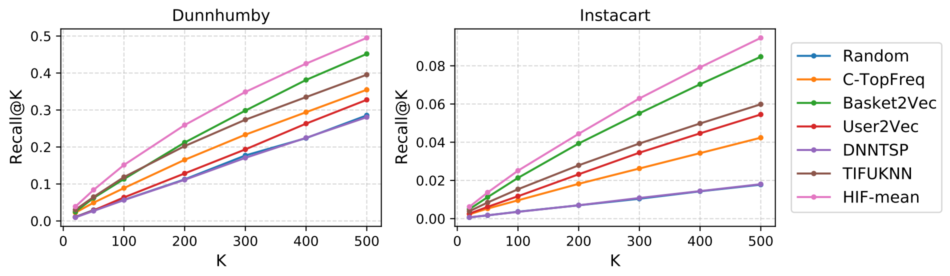
[15]It is not meaningful to evaluate the Recall@K for large values of K in the Rep-RNPR task, as the number of repeat user candidates is limited.

Table 7. Mixed-RNPR results of the REUR algorithm and the baselines. Boldface and underline indicate the best and the second best performing model, respectively. Significant improvements of REUR over the best performing baseline results are marked with † (paired t-test, $p < 0.05$). ▲% shows the improvements of REUR against the best performing baseline.

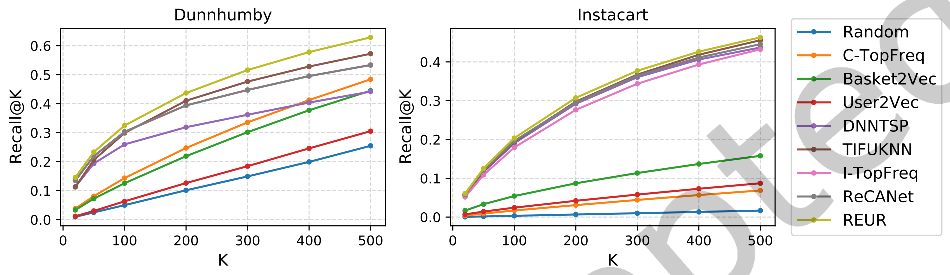
Dataset	Period	Metric	Random	I-TopFreq	C-TopFreq	Basket 2Vec	User 2Vec	DNNTSP	ReCANet	TIFUKNN	REUR (▲%)
Instacart	5	Recall@100	0.0034	0.1794	0.0170	0.0542	0.0243	0.1904	0.1933	0.1941	**0.2037**†(4.9)
		nDCG@100	0.0158	0.5490	0.0650	0.2332	0.1223	0.5933	0.5983	0.5999	**0.6270**†(5.9)
		IHN@100	1.5835	50.8376	6.2624	20.7245	10.9495	54.7954	55.2954	55.4311	**57.9105**†(4.5)
		Recall@200	0.0066	0.2764	0.0309	0.0871	0.0420	0.2920	0.2965	0.2982	**0.3074**†(3.1)
		nDCG@200	0.0162	0.4929	0.0630	0.2030	0.1098	0.5288	0.5316	0.5360	**0.5568**†(3.9)
		IHN@200	3.1285	83.8501	11.5067	34.6381	19.2747	89.7975	90.1346	91.0450	**94.3099**†(3.6)
	10	Recall@100	0.0033	0.1409	0.0163	0.0448	0.0208	0.1446	0.1463	0.1477	**0.1560**†(5.6)
		nDCG@100	0.0238	0.6511	0.0956	0.2829	0.1500	0.6794	0.6809	0.6871	**0.7206**†(4.9)
		IHN@100	2.3681	61.5801	9.2279	25.6278	13.6983	64.0579	64.2261	64.8748	**68.1811**†(5.1)
		Recall@200	0.0065	0.2262	0.0294	0.0736	0.0369	0.2328	0.2356	0.2380	**0.2471**†(3.8)
		nDCG@200	0.0239	0.5685	0.0889	0.2434	0.1341	0.5913	0.5946	0.5999	**0.6252**†(4.2)
		IHN@200	4.7845	104.8940	16.9723	43.7287	24.6122	108.8507	109.6201	110.6156	**115.1574**†(4.1)
Dunnhumby	5	Recall@50	0.0254	0.2190	0.0810	0.0726	0.0302	0.1937	0.2187	0.2054	**0.2333**†(6.5)
		nDCG@50	0.0250	0.2612	0.0774	0.0772	0.0310	0.2271	0.2613	0.2234	**0.2821**†(8.0)
		IHN@50	0.9308	7.5279	2.6521	2.7403	1.2303	6.7633	7.5267	6.8094	**8.0548**†(7.0)
		Recall@100	0.0508	0.3025	0.1436	0.1259	0.0633	0.2597	0.3022	0.2996	**0.3248**†(7.4)
		nDCG@100	0.0357	0.2910	0.1051	0.0990	0.0456	0.2505	0.2907	0.2615	**0.3150**†(8.2)
		IHN@100	1.8140	10.8553	4.7957	4.7572	2.5121	9.7879	10.8014	10.4737	**11.7091**†(7.9)
	10	Recall@50	0.0256	0.2037	0.0758	0.0677	0.0292	0.1784	0.2042	0.1921	**0.2171**†(6.6)
		nDCG@50	0.0356	0.3162	0.1018	0.1015	0.0427	0.2746	0.3166	0.2723	**0.3402**†(7.5)
		IHN@50	1.6067	11.7383	4.3820	4.3947	2.0023	10.4649	11.7391	10.6945	**12.5645**†(7.0)
		Recall@100	0.0496	0.2876	0.1369	0.1206	0.0620	0.2420	0.2870	0.2868	**0.3089**†(7.4)
		nDCG@100	0.0451	0.3280	0.1262	0.1191	0.0571	0.2802	0.3274	0.2977	**0.3536**†(7.8)
		IHN@100	3.1421	17.3898	8.0764	7.8467	4.1466	15.4814	17.3895	16.8825	**18.7001**†(7.5)

For the Mixed-RNPR task, the performance of REUR is shown in Table 7 and Figure 4b. REUR achieves the best performance on both datasets through the combination of *repeat users* ranking (derived from a simple time-aware frequency model) and *explore users* ranking (derived from the HIF model). The improvements range from 3.1% to 8.2% over the best baseline methods. Besides achieving the best performance, an important advantage of the REUR algorithm is that we can explicitly investigate the trade-off between repetition and exploration, which will be discussed in Section 8.4. Note that REUR decouples repetition and exploration, which allows it to benefit from the candidate filtering part by only considering *repeat users*, which we will discuss later in Section 8.5.3.

To sum up, the HIF model, SimpleTF model and REUR algorithm are the state-of-the-art methods on the Expl-RNPR, Rep-RNPR and Mixed-RNPR tasks, respectively.



(a) Expl-RNPR task.



(b) Mixed-RNPR task.

Fig. 4. The Recall@K performance of various methods with K ranging from 20 to 500.

8.3 RQ3: Ablation study of HIF

To analyze the effectiveness of our proposed training strategies and components, we conduct an ablation study on the two datasets. Specifically, we compare the performance of the full HIF model with the following four settings:

(1) No habits module (HIF w/o hab).
(2) No positive augmentation strategy (HIF w/o aug-pos).
(3) No negative adjustment strategy (HIF w/o adj-neg).
(4) No postive augmentation strategy and no negative adjustment strategy (HIF w/o aug-pos and adj-neg).

The results of the ablation study are shown in Table 8. The results show that both the habits capturing module and two strategies are beneficial for the HIF because removing any of them will lead to a decrease in performance. Without habits module, the performance of HIF decreases, ranging from 5% to 20%, which indicates that the frequency information is valuable and the designed habits module is able to leverage this information to model users' shopping habits. Furthermore, HIF w/o hab still outperforms the Basket2Vec baseline (in Table 5), which indicates that the interest module of HIF can capture users' dynamic interests by modeling the dynamic user preferences.

When we employ the positive augmentation strategy, *repeat users* will be sampled and truncated for training. Without positive augmentation, the performance of HIF drops significantly on the Instacart dataset w.r.t. all metrics, ranging from 4.1% to 10.4%, while the drop is not significant in terms of several metrics, i.e., Recall@50, nDCG@50 and IHN@50, on the Dunnhumby dataset. A plausible reason for this result is that the Instacart dataset has more users to be ranked, which means that the Expl-RNPR task is more difficult on Instacart dataset, so HIF can benefit more from the augmented postive samples on the Instacart dataset but the original training samples are enough for finding the top-50 users on the Dunnhumby dataset. Training HIF without negative adjustment strategy results in 4.7% to 21.1% drops in performance. This indicates that avoiding the overlap

Table 8. Expl-RNPR ablation study results. Boldface indicates the best performing model, i.e., HIF with average pooling. Significant deteriorations compared to HIF are marked with ‡ (paired t-test, $p < 0.05$). ▼% shows the drop in performance compared to HIF.

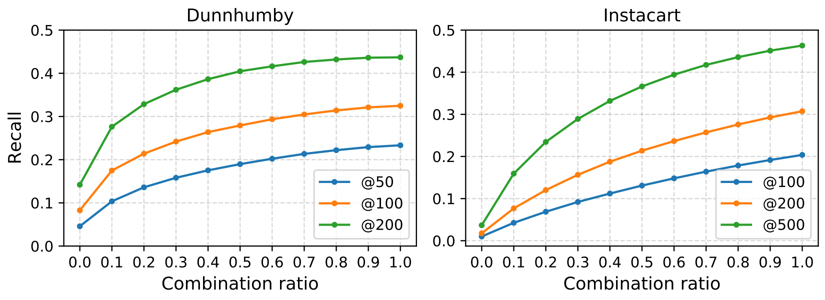
Dataset	Metric	HIF	w/o hab (▼%)	w/o aug-pos (▼%)	w/o adj-neg (▼%)	w/o aug-pos adj-neg(▼%)
Instacart	Recall@100	**0.0251**	0.0237‡(5.9)	0.0233‡(7.2)	0.0198‡(21.1)	0.0206‡(17.9)
	nDCG@100	**0.0346**	0.0311‡(10.1)	0.0310‡(10.4)	0.0282‡(18.5)	0.0289‡(16.5)
	IHN@100	**2.9315**	2.6538‡(9.5)	2.6365‡(10.1)	2.3985‡(18.2)	2.4994‡(14.7)
	Recall@200	**0.0444**	0.0422‡(5.0)	0.0426‡(4.1)	0.0363‡(18.2)	0.0390‡(12.2)
	nDCG@200	**0.0427**	0.0394‡(7.7)	0.0401‡(6.1)	0.0350‡(18.0)	0.0369‡(13.6)
	IHN@200	**5.2537**	4.8525‡(7.6)	4.9452‡(5.9)	4.4325‡(15.6)	4.7102‡(10.3)
Dunnhumby	Recall@50	**0.0841**	0.0718‡(14.6)	0.0833 (1.0)	0.0791‡(5.9)	0.0808‡(3.9)
	nDCG@50	**0.0600**	0.0480‡(20.0)	0.0589 (1.8)	0.0565‡(5.8)	0.0583‡(2.8)
	IHN@50	**1.3001**	1.1472‡(11.8)	1.2984 (0.1)	1.2388‡(4.7)	1.2919 (0.6)
	Recall@100	**0.1514**	0.1281‡(15.4)	0.1456‡(3.8)	0.1376‡(9.1)	0.1436‡(5.1)
	nDCG@100	**0.0877**	0.0710‡(19.0)	0.0845‡(3.6)	0.0806‡(8.1)	0.0832‡(5.1)
	IHN@100	**2.3643**	2.0351‡(13.9)	2.2834‡(3.4)	2.1659‡(8.4)	2.2777‡(3.6)

between training input and prediction input is important and effective in training the HIF model for the Expl-RNPR task. Interestingly, training HIF without both sampling strategies (HIF w/o aug-pos and adj-neg) achieves better performance than training without negative adjustment strategy. The positive augmentation strategy can help us generate more positive samples, which means that the HIF model will also leverage more negative samples during training, and this will increase the probability of using the historical data of the ground-truth users. As a result, negative adjustment is especially important when using positive augmentation strategy, and HIF w/o adj-neg is inferior to other variants of HIF.

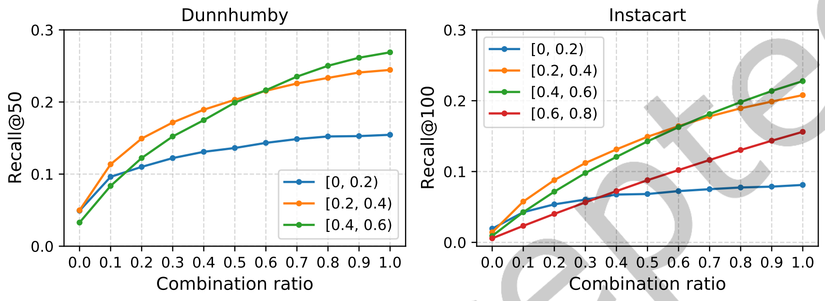
8.4 RQ4: Trade-off analysis for Mixed-RNPR

To investigate the trade-off between repetition and exploration in the Mixed-RNPR task, we use the proposed REUR algorithm and sweep its combination (repeat) ratio α. Figure 5a shows the repetition and exploration trade-off on different datasets when using the same combination ratio α. As the proportion of *repeat users* increases in the recommendation, the performance of REUR on the Mixed-RNPR task increases. REUR achieves its best performance when the combination ratio is 1.0, which indicates that REUR has more confidence in the last user in the *repeat user* ranking than in the first user in the *explore user* ranking. This suggests that the repetition task is much easier than the exploration task in the "item-centered" recommendation that we consider in this paper.

As mentioned before, the repurchase tendency RT of an item can potentially influence the trade-off. To further understand the repetition and exploration trade-off, we also investigate this trade-off on different groups items. To ensure there are enough items within the group, we explore three groups of items, i.e, $RT \in [0, 0.2)$, $[0.2, 0.4)$, $[0.4, 0.6)$ on the Dunnhumby dataset and four groups $RT \in [0, 0.2)$, $[0.2, 0.4)$, $[0.4, 0.6)$, $[0.6, 0.8)$ on the Instacart dataset. The results are shown in Figure 5b. Interestingly, we observe the same trend in all groups of items, the performance increases when the repeat ratio α increases, even in the group with a very low repurchase tendency, i.e., $RT \in [0, 0.2)$. This result once again confirms the large gap in difficulty between the repetition task and the exploration task.



(a) Overall analysis.



(b) Group-level analysis.

Fig. 5. Repetition-exploration trade-off analysis.

8.5 RQ5: Candidate filtering

In this subsection, we first evaluate the effectiveness of candidate filtering, and then discuss the influence on the Expl-RNPR task and insights on the Mixed-RNPR task.

8.5.1 Effectiveness of candidate filtering. To reduce the computational costs at inference time, the candidate filtering process selects a subset of users from the whole user set, which might remove some of the ground-truth users. A good candidate filtering strategy should reduce computational costs, i.e., exclude users who have a low purchase probability w.r.t. the given item, while keeping a large proportion of the ground-truth users *PoG*, i.e., retain as many ground-truth users as possible. Therefore, we analyze the proportion of ground-truth users *PoG* among the candidate users that are left after filtering to evaluate their performance, that is:

$$PoG = \frac{1}{|I^t|} \sum_{i \in I^t} \frac{|T_i \cap \hat{U}_i^t|}{|T_i|},$$

where I^t is a set of items we want to recommend, T_i denotes the ground-truth users, $\hat{U}_i^t$ denotes the set of candidate users for item i after candidate filtering. Note that the computation CP represents a percentage of the original computational costs, that is:

$$CP = \frac{\sum_{i \in I^t} |U_i^t|}{\sum_{i \in I^t} |\hat{U}_i^t|}.$$

Intuitively, we expect the computational costs to increase linearly with the number of item-user scores the model needs to calculate. To validate this, we conduct experiments to investigate the correlation between actual

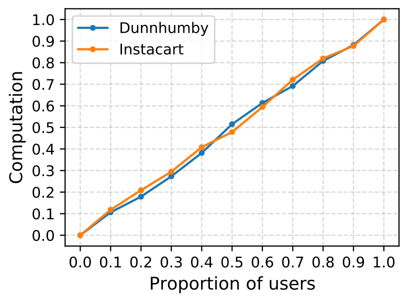


Fig. 6. The relation of actual computation cost and proportion of users that HIF model compute.

Table 9. The proportion of ground-truth users (PoG) and computational costs (CP) of RRBF.

Methods	RRBF-cat for Expl-RNPR		RRBF-item for Mixed-RNPR	
Dataset	*PoG*	Computation (*CP*)	*PoG*	Computation (*CP*)
Instacart	77.70%	60.08%	61.11%	4.5%
Dunnhumby	87.54%	76.12%	43.74%	11.3%

inference times and the defined computational costs *CP*. As shown in Figure 6, a linear correlation between the actual inference times and *CP* is generally observed, and we believe that the minor deviations are a result of the varying lengths of users' historical baskets.[16] Table 9 shows the experimental results for the two repetition rule-based candidate filtering methods introduced in Section 6, i.e., RRBF-cat and RRBF-item. Theoretically, the *PoG* of a random candidate filtering strategy, i.e., randomly selecting a subset of target users as candidates, should be proportional to its computational cost *CP*. Clearly, both RRBF-cat and RRBF-item are effective, since they have a higher left proportion of ground-truth users *PoG* in the candidate user set than using a random strategy. RRBF-cat reaches 77.7% and 87.54% w.r.t. Expl-RNPR *PoG* on Instacart and Dunnhumby, respectively. This result indicates that a large proportion of ground-truth *explore users*, who will explore the given item in the next period, should have already purchased some other items within the given item's category. RRBF-item retains a high proportion of ground-truth users *PoG* w.r.t. Mixed-RNPR on both datasets, while it only has 4.5% and 11.3% of the original computational costs on the Instacart and Dunnhumby datasets, respectively.

In practice, if the prediction results of the CFM model cannot be reused in other tasks in the platform, the computational costs of the CFM model should also be factored into the item-centered recommendation computational costs. We also conduct experiments to compare the actual inference time of the HIF model and CFM model. We find that the average inference times for the CFM model to compute one user-category score is 0.15ms on the Instacart dataset and 0.21ms on the Dunnhumby dataset, whereas the average inference time for the HIF model to compute one item-user score is 3.3ms on the Dunnhumby dataset and 2.1ms on the Instacart dataset. Considering that the number of categories is limited and the CFM model is much lighter than the HIF model, we also evaluated the actual computational costs compared to the original computational costs without candidate filtering and find that the total inference time of the CFM model constitutes only 0.31% and 0.78% of the original inference time on the Instacart and Dunnhumby datasets, respectively. Figure 7 and 8 show the results of model-based candidate filtering (CFM) on the Expl-RNPR task. As the filtering threshold increases, both the computation *CP* and the left proportion of ground-truth users *PoG* decreases, since more ground-truth users are removed from the candidate set with a higher filtering threshold. We can also observe that CFM has a

[16]The computation cost is normalized by the total computation cost by using all users as candidates for each item.

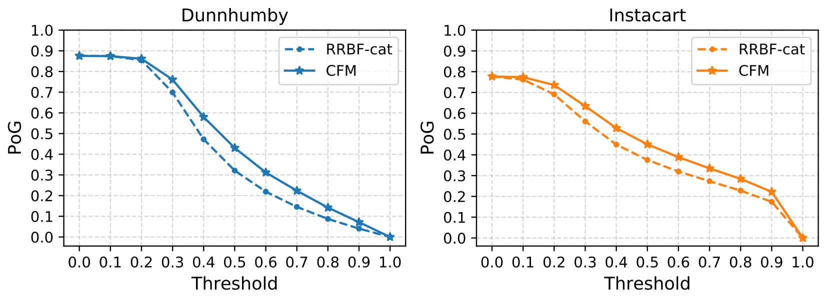


Fig. 7. Proportion of ground-truth users with the CFM filtering threshold changing from 0 to 1. RRBF-cat represent the random strategy within the RRBF-cat candidates.

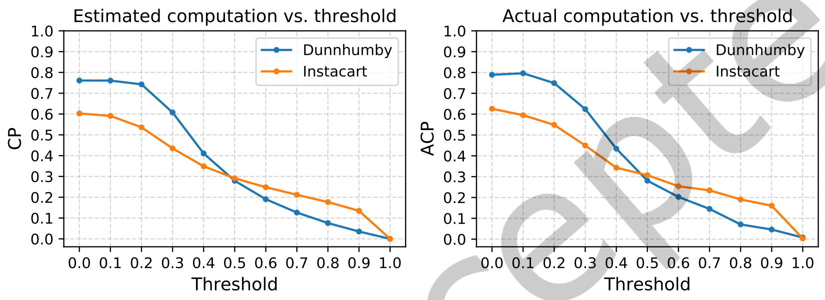


Fig. 8. Computational costs (left) and actual computational cost (right) with the CFM filtering threshold changing from 0 to 1.

higher Expl-RNPR *PoG* than using a random strategy within the candidate set of RRBF-cat in both datasets. This result indicates that CFM can further filter candidates effectively on top of RRBF-cat by considering temporal information.

8.5.2 Influence on the Expl-RNPR task. To understand the influence of candidate filtering on the performance of HIF for the Expl-RNPR task, we analyze the sensitivity of the performance of HIF w.r.t. the candidate filtering threshold of CFM. The experimental results are shown in Figure 9. Note that the performance of HIF when using RRBF-cat is equal to the performance when using CFM with the filtering threshold 0, since they have the same set of candidates. We can observe that HIF with RRBF-cat can achieve same or even higher performance than computing on all target users, while HIF with RRBF-cat only needs 60.08% and 76.12% of the original computational costs on Instacart and Dunnhumby, respectively.

Increasing the CFM filtering threshold leads to a decrease in computational costs (see Figure 9), however, the performance of HIF remains at the same level as RRBF-cat until 0.5, and then decreases gradually as there are fewer candidate users left for each item. This result suggests that the CFM model can help HIF to remove a lot of users who have low probability of purchasing a given item by considering category-level repurchase behavior. With 0.5 as candidate filtering threshold, the CFM model is able to further reduce by 50% (Instacart) and 60.5% (Dunnhumby) the computational costs on top of RRBF-cat, while achieving the same level of performance as using RRBF-cat candidates or, indeed, using all users.

8.5.3 Insights on the Mixed-RNPR task. The trade-off analysis in Section 8.4 suggests that the repetition task is much easier than the exploration task, and that *repeat users* dominate the final recommendation set in REUR.

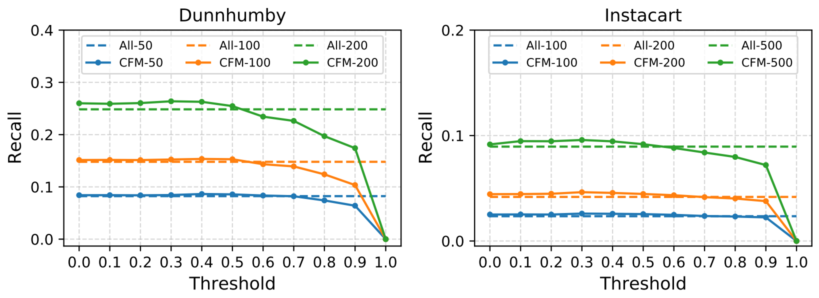


Fig. 9. The Recall performance of HIF with the CFM filtering threshold changing from 0 to 1. The dashed line is the performance of using all target users.

Table 9 shows that only considering the *repeat users*, i.e., RRBF-item, can substantially reduce the computational costs. Besides, the repetition task can be solved by a simple time-aware frequency model, however the exploration task requires a model to infer users' interests in the given item by using item-item correlations and complex representations. Considering the above facts, it is reasonable to ask: can we ignore *explore users* and only consider *repeat users* in the Mixed-RNPR task? The answer might be different depending on different assumptions: (i) If there is no special demand for addressing the need of *explore users* and only focusing on optimizing the accuracy aspect, we believe that reducing the Mixed-RNPR task to the Rep-RNPR task, which simply recommends users who have purchased the given item before is an efficient and effective solution.[17] We believe this is an important point to make, even though this makes the Mixed-RNPR a less challenging algorithmic problem. (ii) An important goal of a recommender system is to connect items with new customers, if beyond accuracy metrics need to be considered, e.g., the long-term recommendation effect, the explore users should not be ignored and we should consider the balance between repeat users and explore users in the recommendation.[18]

9 CONCLUSION

In this paper, we have studied an item-centered sequential recommendation problem, i.e., reverse next-period recommendation (RNPR), which aims to help a given item find top-k users in the next period. We have introduced three-subtasks for RNPR, i.e., Expl-RNPR, Rep-RNPR and Mixed-RNPR, considering the differences in types of target users. For the Expl-RNPR task, we propose a habit-interest fusion model (HIF), which leverages frequency information to model users' habits and pre-trained embeddings to model users' interests. For training the HIF model, we propose two strategies to construct training samples, i.e., a positive augmentation strategy and a negative adjustment strategy, to construct training samples. For the Rep-RNPR task, we employ a simple time-aware frequency model, which only uses the users' direct interactions with the given item. For the Mixed-RNPR task, we propose a repetition-exploration ranking user (REUR) algorithm to decouple the repetition task and exploration task, and generate the final ranking by combining two ranked lists. In addition, we also examined how to reduce the computational costs of our approaches without losing performance. Specifically, we proposed two repetition-rule based filtering methods, i.e., RRBF-cat and RRBF-item, and a model-based candidate filtering method (CFM) to further reduce the computational costs of the HIF model during inference.

[17]For instance, some bread might be close to its "best by date", and the supermarket may want to sell the bread without caring whom they sell it to.

[18]For instance, the supermarket wants to promot a certain product, so the item-centered recommender model not only need to find repeat users, which is important for the short-term profit, but also find potential new users, who might account for long-term profit.

Main findings and implications. We have performed extensive experiments on two publicly available grocery shopping datasets and the experimental results demonstrate the effectiveness of our proposed methods and strategies. The repetition analysis shows that people have stable repetition behavior at the category level in grocery shopping, which is a strong indicator that can be used to find potential top-k users for a given item. Besides, our experiments further show that filtering out some target users using candidate filtering methods, i.e., RRBF and CFM, can effectively reduce computational costs without sacrificing performance.

Apart from proposing solutions, we have also investigated the performance of state-of-the-art user-centered NBR models on the RNPR problem and found that their performance cannot always be generalized to the RNPR task, even though they are good at helping users find top-k items. This result suggests that we should not directly use user-centered recommendation (NBR) methods for the item-centered recommendation (RNPR) task, and that task-specific algorithms should be designed to cater for the RNPR task.

With the proposed REUR algorithm, we also investigated the trade-off between repetition and exploration in Mixed-RNPR. We found that the repetition task, i.e., finding repeat users, is much easier than the exploration task, i.e., finding explore users, in item-centered sequential recommendation, and only recommending *repeat users* is an effective and efficient approach for the Mixed-RNPR task. This imbalance in difficulty can also be found in the user-centered NBR task [25]. A broader implication of this finding is that it is necessary to consider and investigate the differences and trade-offs between repetition and exploration in various recommendation scenarios.

Limitations. Despite the effectiveness of the proposed methods, one limitation of the HIF model is that it cannot be applied to find users for cold-start items currently. Even though we avoid using item-specific trainable parameters in the HIF model, it is still difficult to derive meaningful representations for cold-start items with limited interactions. One possible solution for cold-start items in the RNPR task is to employ a cold-start item representations learning method [see, e.g., 36].

Future work. Work on the RNPR task can be extended in a number of ways. For simplicity, we have employed a simple time-aware frequency model for the repetition task in this paper. We intend to consider item characteristics and correlations to further improve the repetition performance. Second, we have used a conventional skip-gram algorithm to obtain pre-trained item embeddings in this paper. Instead, we want to try and use recent graph neural networks to learn better representations for RNPR as a potential future work. Third, different items might have different numbers of potential users in the next period, so it is interesting to address the RNPR task with a dynamic number of users to recommend. Fourth, we have concentrated on maximizing accuracy, specifically in identifying the correct users for a given item, without considering the potential impact of recommending either repeat users or explore users. It would be interesting to examine the differences in the causal effect between repetition and exploration and to contemplate a way to achieve a balance between them. Finally, we have only focused on the RNPR problem itself in this paper; it is of interest to investigate additional dimensions, such as its potential influence on user satisfaction or fairness among items.

REPRODUCIBILITY

To facilitate the reproducibility of the reported results, this work only uses publicly available data and our source code is publicly available at https://github.com/liming-7/RNPR-Rec.

ACKNOWLEDGMENTS

We would like to thank our reviewers and associate editor for their constructive feedback. This research was partially supported by Ahold Delhaize, the China Scholarship Council (grant #20190607154), and the Hybrid

Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Daichi Amagata and Takahiro Hara. 2021. Reverse Maximum Inner Product Search: How to Efficiently Find Users who Would Like to Buy My Item?. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 273–281.

[2] Mozhdeh Ariannezhad, Sami Jullien, Ming Li, Min Fang, Sebastian Schelter, and Maarten de Rijke. 2022. ReCANet: A Repeat Consumption-Aware Neural Network for Next Basket Recommendation in Grocery Shopping. In *SIGIR 2022: 45th international ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1240–1250.

[3] Mozhdeh Ariannezhad, Ming Li, Sebastian Schelter, and Maarten de Rijke. 2023. A Personalized Neighborhood-based Model for Within-basket Recommendation in Grocery Shopping. In *WSDM 2023: The Sixteenth International Conference on Web Search and Data Mining*. ACM, 87–95.

[4] Ting Bai, Jian-Yun Nie, Wayne Xin Zhao, Yutao Zhu, Pan Du, and Ji-Rong Wen. 2018. An Attribute-aware Neural Attentive Model for Next Basket recommendation. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1201–1204.

[5] Oren Barkan and Noam Koenigstein. 2016. Item2vec: Neural Item Embedding for Collaborative Filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. 1–6.

[6] Rahul Bhagat, Srevatsan Muralidharan, Alex Lobzhanidze, and Shankar Vishwanath. 2018. Buy It Again: Modeling Repeat Purchase Recommendations. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 62–70.

[7] Oscar Celma. 2010. Music Recommendation. In *Music Recommendation and Discovery*. Springer, 43–85.

[8] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-decoder Approaches. *arXiv preprint arXiv:1409.1259* (2014).

[9] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. 2010. The YouTube Video Recommendation System. In *Proceedings of the 4th ACM conference on Recommender systems*. 293–296.

[10] Mukund Deshpande and George Karypis. 2004. Item-based Top-n Recommendation Algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[11] Guglielmo Faggioli, Mirko Polato, and Fabio Aiolli. 2020. Recency Aware Collaborative Filtering for Next Basket Recommendation. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*. 80–87.

[12] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in Your Inbox: Product Recommendations at Scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1809–1818.

[13] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 843–852.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06939* (2015).

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (1997), 1735–1780.

[16] Haoji Hu and Xiangnan He. 2019. Sets2Sets: Learning from Sequential Sets with Neural Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1491–1499.

[17] Haoji Hu, Xiangnan He, Jinyang Gao, and Zhi-Li Zhang. 2020. Modeling Personalized Item Frequency Information for Next-basket Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1071–1080.

[18] Xiaowen Huang, Quan Fang, Shengsheng Qian, Jitao Sang, Yan Li, and Changsheng Xu. 2019. Explainable Interaction-driven User Modeling over Knowledge Graph for Sequential Recommendation. In *Proceedings of the 27th ACM International Conference on Multimedia*. 548–556.

[19] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored Item Similarity Models for Top-n Recommender Systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 659–667.

[20] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. 197–206.

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[22] Duc-Trong Le, Hady W Lauw, and Yuan Fang. 2019. Correlation-sensitive Next-basket Recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2808–2814.
[23] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. 2016. On the Value of Reminders within E-commerce Recommendations. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. 27–35.
[24] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.
[25] Ming Li, Sami Jullien, Mozhdeh Ariannezhad, and Maarten de Rijke. 2023. A Next Basket Recommendation Reality Check. *ACM Transactions on Information Systems* 41, 4 (2023), Article 116.
[26] Ming Li, Ali Vardasbi, Andrew Yates, and Maarten de Rijke. 2023. Repetition and Exploration in Sequential Recommendation: A Reproducibility Study. In *SIGIR 2023: 46th international ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.
[27] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized News Recommendation Based on Click Behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*. 31–40.
[28] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term Attention/Memory Priority Model for Session-based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.
[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).
[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. *Proceedings of the 43rd Advances in Neural Information Processing Systems* (2013).
[31] Yuqi Qin, Pengfei Wang, and Chenliang Li. 2021. The World is Binary: Contrastive Learning for Denoising Next Basket Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 859–868.
[32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 811–820.
[33] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.
[34] V Subramaniyaswamy, Gunasekaran Manogaran, R. Logesh, V. Vijayakumar, Naveen Chilamkurti, D. Malathi, and N. Senthilselvan. 2019. An Ontology-driven Personalized Food Recommendation in IoT-based Healthcare System. *The Journal of Supercomputing* 75, 6 (2019), 3184–3216.
[35] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450.
[36] Jacopo Tagliabue, Bingqing Yu, and Federico Bianchi. 2020. The Embeddings that Came in from the Cold: Improving Vectors for New and Rare Products with Content-based Inference. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 577–578.
[37] Pei Jie Tan, Arry Tanusondjaja, Armando Corsi, Larry Lockshin, Christopher Villani, and Svetlana Bogomolova. 2022. Audit and Benchmarking of Supermarket Catalog Composition in Five Countries. *International Journal of Advertising* (2022), 1–28.
[38] Jiaxi Tang and Ke Wang. 2018. Personalized Top-n Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 565–573.
[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *Advances in Neural Information Processing Systems* 30 (2017).
[40] Akrivi Vlachou, Christos Doulkeridis, Yannis Kotidis, and Kjetil Nørvåg. 2010. Reverse Top-k Queries. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. 365–376.
[41] Akrivi Vlachou, Christos Doulkeridis, Yannis Kotidis, and Kjetil Norvag. 2011. Monochromatic and Bichromatic Reverse Top-k Queries. *IEEE Transactions on Knowledge and Data Engineering* 23, 8 (2011), 1215–1229.
[42] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. 2018. Representing and Recommending Shopping Baskets with Complementarity, Compatibility and Loyalty. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1133–1142.
[43] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-aware Network for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference*. 1835–1844.
[44] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for Next Basket Recommendation. In *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*. 403–412.
[45] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Defu Lian. 2021. A Survey on Session-based Recommender Systems. *ACM Computing Surveys (CSUR)* 54, 7 (2021), 1–38.
[46] Shiqi Wang, Chongming Gao, Min Gao, Junliang Yu, Zongwei Wang, and Hongzhi Yin. 2022. Who Are the Best Adopters? User Selection Model for Free Trial Item Promotion. *arXiv preprint arXiv:2202.09508* (2022).

[47] Shoujin Wang, Liang Hu, Yan Wang, Quan Z Sheng, Mehmet Orgun, and Longbing Cao. 2020. Intention Nets: Psychology-inspired User Choice behavior Modeling for Next-basket Prediction. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. 6259–6266.

[48] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. 5329–5336.

[49] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[50] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-encoders for Top-n Recommender Systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. 153–162.

[51] Fulya Yalvaç, Veranika Lim, Jun Hu, Mathias Funk, and Matthias Rauterberg. 2014. Social Recipe Recommendation to Reduce Food Waste. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*. 2431–2436.

[52] Ghim-Eng Yap, Xiao-Li Li, and Philip S Yu. 2012. Effective Next-Items Recommendation via Personalized Sequential Pattern Mining. In *Proceedings of the International Conference on Database Systems for Advanced Applications*. 48–64.

[53] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 729–732.

[54] Le Yu, Leilei Sun, Bowen Du, Chuanren Liu, Hui Xiong, and Weifeng Lv. 2020. Predicting Temporal Sets with Deep Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1083–1091.

[55] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 582–590.

[56] Zhao Zhang, Cheqing Jin, and Qiangqiang Kang. 2014. Reverse k-ranks Query. *Proceedings of the VLDB Endowment* 7, 10 (2014), 785–796.