TTRS: Tinkoff Transactions Recommender System benchmark

Sergey Kolesnikov, Oleg Lashinin, Michail Pechatov, Alexander Kosov
Tinkoff
{s.s.kolesnikov, o.a.lashinin, m.pechatov, a.kosov}@tinkoff.ai
Russia

ABSTRACT
Over the past decade, tremendous progress has been made in inventing new RecSys methods. However, one of the fundamental problems of the RecSys research community remains the lack of applied datasets and benchmarks with well-defined evaluation rules and metrics to test these novel approaches. In this article, we present the TTRS - Tinkoff Transactions Recommender System benchmark. This financial transaction benchmark contains over 2 million interactions between almost 10,000 users and more than 1,000 merchant brands over 14 months. To the best of our knowledge, this is the first publicly available financial transactions dataset. To make it more suitable for possible applications, we provide a complete description of the data collection pipeline, its preprocessing, and the resulting dataset statistics. We also present a comprehensive comparison of the current popular RecSys methods on the next-period recommendation task and conduct a detailed analysis of their performance against various metrics and recommendation goals. Last but not least, we also introduce Personalized Item-Frequencies-based Model (Re)Ranker – PIFMR, a simple yet powerful approach that has proven to be the most effective for the benchmarked tasks.

CCS CONCEPTS
• Computer systems organization → Embedded systems; Redundancy; Robotics; • Networks → Network reliability.

KEYWORDS
datasets, neural networks, recommender systems

1 INTRODUCTION
In the current era of big data and global personalization, Recommender Systems (RecSys) are playing a pivotal role in improving user experience in a large variety of domains: movies [43], music [35], news [12] and many more [5, 34, 55]. There are many different types of recommendations, such as rating prediction [23], as well as top-n [9], sequential [54], session-based [53], and next-basket [16, 37, 40] recommendations. Thanks to close collaboration between academia and industry professionals [3, 7] on joint research topics, many new articles and methods are released every year [58]. Various new techniques aim to improve RecSys approaches with deep learning-based methods [28, 37, 56], memory-based methods [42], latent factor-based methods [9, 24, 38], or reinforcement learning [1, 29]. With such a rapid growth of new approaches, it is almost impossible to evaluate all of them correctly. Moreover, each application requires its own logic for data preprocessing and evaluation setup [32]. Last but not least, current machine learning methods require careful selection of hyperparameters, which only complicates evaluation correctness [4, 50].

To combat the challenges above, many RecSys benchmark frameworks have been created: RecBole [60], Elliot [2], ReChorus [51], DaisyRec [47]. All these benchmarks cover such popular tasks as top-n, next-item, or next-basket recommendations, which are essential for real-world applications. However, other equally valuable scenarios have received little attention in the research community. In our work, we focus on one of these scenarios. Specifically, we consider forming personalized recommendations for a certain period – the next-period recommendation task [59]. This task can have various business applications, such as monthly customers’ cashback recommendations or products-of-interest list creation during users’ online sessions on websites [48].

To this end, we would like to propose a large-scale financial transactions dataset and provide a comprehensive comparison of the current popular RecSys methods on the next-period recommendation task. The contributions of this paper are summarized as follows:

• Our first contribution is a large-scale financial transactions dataset – TTRS (Section 3). It contains over 2 million interactions between almost 10,000 users and more than 1,000 merchants for a total of 14 months. To the best of our knowledge, it is the first publicly available dataset of financial transactions.

• Our second contribution is benchmarking a various number of existing RecSys methods on the next-period recommendation task (Section 4). To ensure the reproducibility criteria, we carefully describe the evaluation techniques and the metrics required for correct benchmarking. In addition, we conduct a comprehensive quantitative study and comparative analysis of the different methods used in our benchmarking.

• The final point of our study is the proposal of a new approach: Personalized Item-Frequencies-based Model (Re)Ranker, or PIFMR. It summarises our experiments’ findings and achieves the best results on the benchmarked tasks.
We hope that our open-sourced TTRS dataset with a described benchmark on the next-period recommendation task and the proposed PIFMR approach will serve as a foundation for other industrial research labs to develop real-world applied standards for the progress of the RecSys field.

2 RELATED WORK

Large-scale datasets and benchmarks have successively proven their fundamental importance in many research fields. RecSys was no exception, and many recent breakthroughs came with the emergence of such benchmarks [11, 30]. In this section, we observe popular datasets and methods as well as associated benchmarks and evaluation strategies.

2.1 Tasks

In general, the majority of recommender systems aim to learn users’ interests. There are several specific ways we can formulate such recommendation tasks. In the classical formulation, the researchers hide some of the user’s interactions for a test set. Next, the model ranks all items for each user, trying to predict the hidden interactions. This task is called the top-n recommendation task [10]. In another scenario, the model knows the user’s previous ordered history of interactions and predicts the next item the user could interact with. This is the next-item prediction task [36]. If the user can consume sets of items simultaneously and we want to predict the whole set of interactions, this is called the next-basket prediction task [36]. Finally, if we are curious about user interests over time, we could predict their interactions for a predefined period, which is the next-period recommendation task [59]. In this work, we are focused on the former task. Other similar tasks, for example, click-through rate prediction [20], rating prediction [23], are not the focus of our work.

2.2 Datasets

To compare with the transactional nature of TTRS, we reviewed the available RecSys datasets for additional properties: interaction timestamp, transaction amount, and meta-information. These properties are necessary for future studies of the proposed standard. With such requirements, we found two publicly available transaction-based datasets:

- Ta-Feng\(^1\) - a Chinese grocery store dataset that has basket-based transaction data from November 2000 to February 2001. Each transaction has a timestamp. Items with identical timestamps are considered as one basket. This dataset is widely used for next-basket prediction research [14, 16, 25].
- Dunnhumby\(^2\) - a dataset provided by Dunnhumby that contains customers transaction data over a period of 117 weeks from April 2006 to July 2008. For benchmarking purposes, we select the ‘Let’s Get Sort-of-Real sample 50K customers’ version of the dataset, which is well-known among the research community [13, 15, 16].

Statistical information on the raw datasets is summarized in Table 1.

### Table 1: Dataset statistics before and after preprocessing

| Dataset     | Before preprocessing | After preprocessing | Final statistics |
|-------------|----------------------|---------------------|-----------------|
|             | # users | # items | # interactions | # users | # items | # interactions | # inter. per user | # inter. per user | # inter. per item in month | # inter. per item in month |
| TaFeng      | 32,266  | 23,182  | 8,177,41       | 3470     | 2929    | 19,6549       | 56.64            | 14.16             | 67.1                  | 16.78                 |
| Dunnhumby   | 50,000  | 4,997   | 31,057,875     | 11,047   | 3,178   | 11,594,609    | 1049.57          | 40.37             | 3648.4                | 140.32                |
| TTRS        | 50,000  | 2,873   | 14,287,287     | 9,396    | 1,157   | 27,448,287    | 292.13           | 20.87             | 2,372.37              | 169.45                |

\(^1\)https://www.kaggle.com/chiranjividas09/ta-feng-grocery-dataset

\(^2\)https://www.dunnhumby.com/source-files/
researchers highlighted the importance of time-based algorithm validation. In a prior study [47], the authors sampled 85 papers published in 2017-2019 from top conferences and concluded that random-split-by-ratio and leave-one-out splitting strategies are used in 82% cases. At the same time, recent studies [32] pointed out that the most strict and realistic setting for data splitting is a global temporal split, where a fixed time-point separates interactions for training and testing. The authors found that only 2 of 17 recommender algorithms (published from 2009 to 2020) were evaluated using this scenario. In another work [18], the authors compare the impact of data leaks on different RecSys methods. They found that “future data” can improve or deteriorate recommendation accuracy, making the impact of data leakage unpredictable. In this paper, to avoid all the issues above, we use a global temporal K-fold validation scheme (Section 4).

### 3.1 Evaluation Setup

**Data Description.** The crucial part of the TTRS dataset is the diversity of the transaction sources. While other datasets handle only merchant user activity, TTRS contains the whole user financial activity – supermarkets, clothing stores, online delivery shops, cinemas, gas stations, cafes and restaurants, museums, etc. Thus, TTRS contains anonymized information about the daily interests of users based on their transactional activity. To the best of our knowledge, this is the first publicly available dataset that makes it possible to build financial activity recommendations.

**Data Collection.** Our dataset contains transaction information of a randomly selected 50 thousand Tinkoff users from January 2019 to March 2020. Each transaction contains an anonymized user id, transaction type, information about a purchased merchant, transaction timestamp, and transaction amount. Full description of the dataset can be found in Table 2.

### 3.2 Data Preprocessing Pipeline

To prepare the dataset for benchmarking, we apply a few preprocessing steps. First, we truncate the dates to have full months for simplified evaluation on time periods of weeks and months. Secondly, we remove users and items with less than ten interactions in the first six months and filter users with less than one transaction per month to reduce possible anomalies. As the number of interactions between the remaining users and items could change after filtering, we repeat the second step several times until the data converges. Statistical information about clean datasets after preprocessing is summarized in Table 1.

### 4 EXPERIMENTS

For our benchmark, we chose the next-period recommendation task with a period of one month. The main goal of our benchmark was to predict users’ interactions in the next month, using their interaction history over the past few months. In this section, we will go through the experiment setup, benchmarked methods and introduce our main findings and improvements.

#### 4.1 Evaluation Setup

**Metrics.** We compare models with each other using standard metrics widely utilized by researchers: Recall@K, NDCG@K, and MAP@K. Each metric can be calculated for a recommendation list of length K, where K ranges from 1 to the number of items. K is usually called the cutoff, which stands for the length to which the recommendations are cut. We use @10, @20, @50 cutoffs during benchmarking.

**Validation Scheme.** To get the most accurate results, we use several ideas from prior articles [6, 11, 32]. Firstly, we use a global temporal split to separate our training data from test one and prevent possible data leaks associated with seasonal user preferences. Secondly, we use temporal cross-validation with several folds for a more precise model evaluation. Finally, for each such fold with N periods, we use the optimal hyperparameter search through extra data partitioning into "train" (N-2 periods), "validation" (1 period), and "test" (1 period) splits. The best hyperparameters found on the ("train", "validation") split were used to initialize and train the model for final testing on the ("train+validation", "test") split. The entire validation process is shown in Figure 1.

**Hyperparameter Search.** Similar to previous studies [11], we search for the optimal parameters through Bayesian search using the implementation of Scikit-Optimize. For each pair (algorithm, test

### Table 2: TTRS dataset description. We use `party_rk` as user identifier and `merchant_group_rk` as item identifier.

| column             | description                                      | # unique values | column type |
|--------------------|--------------------------------------------------|-----------------|-------------|
| party_rk           | unique user identification (anonymized)          | 50000           | int         |
| financial_account_type_cd | account type                                    | 2               | categorical int |
| transaction_type_desc | transaction type                                | 4               | categorical int |
| merchant_type      | merchant type (anonymized)                      | 464             | categorical int |
| merchant_group_rk  | merchant group identifier (anonymized)           | 2873            | categorical int |
| category           | merchant category                               | 36              | categorical string |
| transaction_dttm   | transaction timestamp                           | -               | datetime    |
| transaction_amt    | transaction amount                              | -               | float       |

---

3The dataset is available on request to authors upon submitting a license agreement.

4https://scikit-optimize.github.io/
Confidence'17, July 2017, Washington, DC, USA
Sergey Kolesnikov, Oleg Lashinin, Michail Pechatov, Alexander Kosov

Figure 1: Evaluation procedure. On step (I), we train model on the (“Train 1”, “Validation 1”) split to find optimal hyperparameters for the first fold. On step (II), we use these hyperparameters to train the model on the (“Train 1 + Validation 1”, “Test 1”) split to evaluate the model on the first fold. On step (III), we repeat the (I) and (II) steps for the next fold.

fold), we iterate over 25 hypotheses, where the first 5 are random. We use MAP@10 metric for model selection. We share all the code as well as details of the respective hyperparameter ranges and the final algorithms’ hyperparameters online.

Computational Resources. The experiments were run on a single machine with NVIDIA Tesla V100, 200GB RAM, and Intel(R) Xeon(R) CPU @ 3.00GHz (16 cores) for 5 days.

4.2 Methods
In this section, we would like to briefly describe approaches used for benchmarking next-period recommendations:

- **TopPopular** [19], **TopPersonal** [16] are simple RecSys baselines that work based on item popularity. The TopPopular algorithm recommends the most popular items, sorting them in descending order of global popularity. The TopPersonal method focuses on items with which the user has already interacted. The recommendation list is created by sorting them in descending order of interaction frequency. If no personal recommendations are found, TopPersonal uses the TopPopular approach for prediction.

- **NMF** [26], **PureSVD** [9], **IALS** [17] are matrix factorization-based (MF-based) models. These models are designed to approximate any value in the interaction matrix by multiplying the user and item vectors in the hidden space. The interaction matrix could be represented as a matrix with interaction frequencies or in binary form. We use binary_matrix hyperparameter to handle such data preprocessing. In the first case, we apply the $log(1 + p)$ transformation. In another case, all frequencies above 0 were replaced by 1.

- **SLIM** [33], **EASE** [45] are linear models that learn an item-item weight matrix. Similar to MF-based models, we add a hyperparameter binary_matrix to approximate frequencies or a binary mask of interactions.

- **Multi-VAE** [28], **Macrid-VAE** [31], **RecVAE** [44] are variational autoencoder approaches (VAE-based) for a top-n recommendation task. They utilize the idea of using multinomial likelihood to recover inputs from hidden representations and use them for recommendations.

- **GRU4Rec** [49], **SASRec** [22], **BERT4Rec** [46] are sequential-based models. Unlike previously mentioned methods, these models know the sequence of users’ interactions and learn sequence representation with RNN- or self-attention-based neural networks. This representation is then used for next-item recommendations.

- **RepeatNet** [37] is an RNN-based model that uses a repeat-explore mechanism for session-based recommendations. The model has two different recommendation modes. In the first, “repeat” mode, the model recommends something from users’ consumption history. In the second, “explore” mode, the model recommends something new that hasn’t been listed in the input sequence.

- **ItemKNN** [41] is an item-based k-nearest neighbors method, which utilizes similarities between previously purchased items. Similar to [11], we used different similarity measures during our experiments: Jaccard coefficient, Cosine, Asymmetric Cosine, and Tversky similarity.

- **TIFUKNN** [16] is a current state-of-the-art method for the next-basket recommendation task. It uses the idea of learning Temporal Item Frequencies with the k-nearest neighbors approach.

To summarize, we consider 16 models, 7 of which are neural networks, and 5 are sequence-aware. For benchmarking purposes, we include approaches of different types such as matrix factorization, linear models, variational autoencoders, recurrent neural networks, and self-attention-based methods.

We adapted all the above models for the next-period recommendation task. To predict the next period (one month in our case), we use all available history for Statistical or General Recommenders.
| Dataset   | Model          | Type   | RECALL | NDCG     | MAP   | Time       |
|-----------|----------------|--------|--------|----------|-------|------------|
| TopPopular| TopPopular     | Statistical | 9.38   | 13.17    | 19.88 |            |
|           | PureSVD        | MF     | 19.14  | 26.06    | 34.56 | 15.28      |
|           | IALS           |        | 11.95  | 18.34    | 29.89 | 21.57      |
|           | SLIM I2I       |        | 21.24  | 30.15    | 41.72 | 15.17      |
| MultiIST  | EASE I2I       | VAE    | 13.13  | 20.1     | 31.96 | 19.96      |
|           | MacridVAE      |        | 21.24  | 30.15    | 41.72 | 30.92      |
|           | RecVAE         |        | 24.13  | 29.69    | 40.76 | 31.17      |
| GRIARec   | Sequence       |        | 10.96  | 15.09    | 22.45 | 17.34      |
| SASRec    | Sequence       |        | 15.13  | 20.42    | 29.06 | 22.95      |
| BERTRec   | Sequence       |        | 15.37  | 20.79    | 29.67 | 23.55      |
| RepeatNet | Sequential     |        | 22.27  | 31.53    | 44.75 | 32.37      |
| ItemKNN   | TopKNN         | KNN    | 10.32  | 22.46    | 34.92 | 16.26      |
|           |                |        | 20.98  | 29.61    | 44.16 | 30.47      |
|           | SLIM I2I       | PIFMR  | 22.68  | 31.62    | 38.85 | 33.05      |
| EASE      |                |        | 22.67  | 32.58    | 46.06 | 33.03      |
| MultiIST  |                | VAE    | 22.66  | 32.54    | 46.05 | 33.03      |
| ItemKNN   |                |        | 16.65  | 24.85    | 36.36 | 24.24      |
| TopPopular| TopPopular     | Statistical | 28.79  | 43.17    | 59.07 | 31.18      |
| MultiIST  | EASE I2I       | VAE    | 50.06  | 67.28    | 83.42 | 47.13      |
| MacridVAE |                |        | 52.64  | 62.87    | 76.15 | 57.09      |
| RecVAE    |                |        | 54.74  | 65.39    | 78.19 | 58.95      |
| GRIARec   | Sequence       |        | 58.65  | 68.97    | 80.14 | 62.66      |
| SASRec    | Sequence       |        | 54.01  | 64.48    | 76.94 | 58.12      |
| BERTRec   | Sequence       |        | 55.86  | 66.68    | 79.60 | 59.03      |
| RepeatNet | Sequential     |        | 61.15  | 72.74    | 82.49 | 63.2       |
| ItemKNN   | TopKNN         | KNN    | 31.81  | 43.32    | 57.78 | 33.9       |
|           |                |        | 59.96  | 71.93    | 81.3  | 62.98      |
| TTRS      |                | PIFMR  | 61.39  | 73.53    | 84.19 | 64.01      |
| EASE      |                |        | 61.31  | 73.43    | 83.94 | 63.96      |
| MultiIST  |                | VAE    | 61.34  | 73.42    | 84.2  | 63.97      |
| ItemKNN   |                |        | 53.82  | 64.97    | 75.8  | 55.77      |
| TopPopular| TopPopular     | Statistical | 4.77   | 8.14     | 14.15 | 3.38       |
| MultiIST  | EASE I2I       | VAE    | 13.11  | 17.13    | 23.89 | 15.38      |
| MacridVAE |                |        | 11.29  | 15.35    | 20.1  | 12.91      |
| RecVAE    |                |        | 12.42  | 17.91    | 24.1  | 13.8       |
| GRIARec   | Sequence       |        | 11.55  | 18.15    | 25.86 | 11.14      |
| SASRec    | Sequence       |        | 11.91  | 17.33    | 25.62 | 12.84      |
| BERTRec   | Sequence       |        | 13.58  | 19.42    | 27.31 | 14.4       |
| RepeatNet | Sequential     |        | 7.91   | 18.87    | 24.38 | 9.45       |
| ItemKNN   | TopKNN         | KNN    | 13.04  | 18.81    | 27.68 | 14.48      |
|           |                |        | 13.05  | 18.55    | 25.98 | 14.76      |
| TaFeng    |                | PIFMR  | 5.06   | 8.78     | 14.95 | 5.14       |
| EASE      |                |        | 10.06  | 14.5     | 22.44 | 10.78      |
| MultiIST  |                | VAE    | 6.84   | 10.07    | 16.55 | 7.87       |
| ItemKNN   |                |        | 7.29   | 10.93    | 14.87 | 7.31       |
|             | TIFUKNN        | KNN    | 13.44  | 19.54    | 27.73 | 15.09      |

**Table 3:** Next-month recommendation benchmark. Ground truth test interactions are not necessarily new to users, and recommendation lists are not filtered in any way. All metrics are averaged over 2 (TaFeng) or 6 (TTRS and Dunnhumby) test folds. Benchmark time is estimated as the overall time required for a hyperparameter search (25 hypotheses) and all included metrics calculations.
Algorithm 1 Personalized Item Frequency Model (Re)Ranker

1: \( \text{MinFreq} \) - hyperparameter
2: \( F \) - personalized frequency-based item statistics
3: \( S \) - user-item scores from the model
4: \( c = 1 - \) normalization constant
5: \( F_{freq} = F > \text{MinFreq} \)
6: \( S_{\max} = \max(S), S_{\min} = \min(S) \)
7: \( S_{\text{top}} = (S - S_{\min} + c)/(S_{\max} - S_{\min} + 2 \times c) \)
8: \( S_{\text{PIFMR}} = S_{\text{top}} + F_{freq} \)

and 100 last interactions for Sequential Recomenders. All models produced predictions in the form of sorted lists of items, and were evaluated on the next-month recommendation task in the same manner.

4.3 Results
The results of the model’s comparison can be found in Table 3. Several observations can be made based on these results. First, the simple baselines based on personalized items’ popularity are compatible with other methods across all datasets. This indicates the importance of the user’s repeat consumption pattern for recommendations.

Second, many current RecSys methods of different types (MF, linear, VAEs, sequential) could hardly beat the TopPersonal baseline across all metrics and datasets. We believe the reason for that is that for these approaches, the repetitive nature of the datasets is too difficult to generalize.

Third, recently proposed deep learning methods, such as RepeatNet, achieve performance comparable with the TopPersonal baseline. We believe the reason is that the “repeat” mode of the proposal network helps generalize the datasets’ repetitive nature.

4.4 Personalized Item-Frequencies-based Model (Re)Ranker
Analyzing benchmark results, it is easy to notice the importance of repetitive user interactions with items. For example, the TopPersonal baseline often gives better results than recent machine learning approaches, where only one method out of 14 could achieve a better MAP@10 score. However, note that TopPersonal also has several disadvantages: (1) it cannot correctly rank items with the same consumption frequency, (2) it is unable to utilize users’ interaction history to identify novel items for recommendation.

To overcome these issues, we propose a simple yet powerful improvement by introducing Personalized Item-Frequencies-based Model (Re)Ranker - PIFMR. Suppose we have a history of user interaction with items, and we can aggregate these interactions into a vector \( F = (f_{u1}, ..., f_{um}) \) where \( f_{ui} \) is the number of times that item \( i \) was purchased by a user \( u \) and \( m \) is the number of items. Recommending new items by these frequencies will bring us to a TopPersonal-like approach. On the other hand, using the same interaction history, we could train a simple model such as EASE, SLIM, MF, or VAE and form their prediction vector as \( S = (s_{u1}, ..., s_{um}) \) where \( s_{ui} \) is a relevance score for a user-item pair \((u, i)\). The main idea of PIFMR is to use Personalized Item Frequencies (vector \( F \)) to re-rank model predictions (vector \( S \)). We also perform a monotonic transformation on the model’s scores so that they lie in the \((0, 1)\) interval. As a result of this, for any pairs of items \( k \) and \( r \) PIFMR gives such predictions \( p_r \) where \( p_{rk} < p_{ur} \) if \( f_{rk} < f_{ur} \). The final algorithm is presented in Algorithm 1. Retrained PIFMR-based models could be found in Table 3, “PIFMR” type.

Combining predictions with a PIFMR-based model, we solved several challenges at once: (1) the model is capable of learning how to rank items with the same consumption frequency, (2) thanks to base model usage, we could find patterns in the users’ behavior and recommend new personalized items, (3) the model easily identifies high-frequency repetitive patterns in consumption history, (4) as well as robust low-frequency purchase “anomalies”, thanks to the \( \text{MinFreq} \) threshold.

4.5 New Pattern Finding Analysis
When it comes to the importance of repeated patterns, it is also interesting to investigate the benchmarked methods’ ability to find new relevant items for users that are not yet present in their history. To do so, we test the same trained models on a slightly different task. Rather than analyze the method’s performance across all possible items, both new and repeated for that user, we measure its efficiency on new items only. The results of the models’ comparison for this recommendation goal can be found in Table 4.

A few interesting observations can be made based on our results. First, while TIFUKNN is showing average results across datasets, it does the best on the TaFeng one. A possible reason for this could be that the amount of data in the TaFeng dataset is rather small compared to the other ones. Second, TTRS is the only dataset where the TopPopular approach still achieves competitive results that could show the similarity of user interests in this dataset. Third, MF-based methods show the worst performance on this task across all datasets. Fourth, RepeatNet performs worse than TopPopular, which may indicate that it overfits to the “repeat” mode rather than “explore”. Lastly, while the proposed PIFMR approach usually slightly worsens the performance of the MultiVAE, EASE, and SLIM methods on this task, it actually improves the performance of the ItemkNN approach (and SLIM on the TaFeng dataset). In addition to the Table 3 results, the above findings may serve as a good reason for further research on the methodology of PIF-based recommendations.

5 CONCLUSION
In this paper, we proposed a large-scale financial transactions benchmark named TTRS that is based on user-merchant interactions. We evaluated various RecSys methods on several transaction-based datasets to compare the effects of different factors on the next-period recommendation task. As shown by the benchmark, the user consumption repeatability factor is ubiquitous in many real-world applications and challenging for current RecSys methods. With this new benchmark, we also presented a simple yet powerful approach: Personalized Item-Frequencies-based Model (Re)Ranker, or PIFMR, which helped in improving the performance of RecSys methods on benchmarked tasks.

REFERENCES
[1] M. Mehdi Afsar, Trafford Crump, and Behrouz H. Far. 2021. Reinforcement learning based recommender systems: A survey. CoRR abs/2101.06286 (2021).
## Table 4: Next-month new item recommendation benchmark. Ground truth objects are new to users, and recommendation lists do not contain items interacted with during training. All metrics are averaged over 2 (TaFeng) or 6 (TTRS and Dummoby) test folds.
