Our Model Achieves Excellent Performance on MovieLens: What Does It Mean?

YU-CHEN FAN, YITONG JI, JIE ZHANG, and AIXIN SUN, Nanyang Technological University, Singapore, Singapore

A typical benchmark dataset for recommender system (RecSys) evaluation consists of user-item interactions generated on a platform within a time period. The interaction generation mechanism partially explains why a user interacts with (e.g., like, purchase, rate) an item, and the context of when a particular interaction happened. In this study, we conduct a meticulous analysis of the MovieLens dataset and explain the potential impact of using the dataset for evaluating recommendation algorithms. We make a few main findings from our analysis. First, there are significant differences in user interactions at the different stages when a user interacts with the MovieLens platform. The early interactions largely define the user portrait which affect the subsequent interactions. Second, user interactions are highly affected by the candidate movies that are recommended by the platform's internal recommendation algorithm(s). Third, changing the order of user interactions makes it more difficult for sequential algorithms to capture the progressive interaction process. We further discuss the discrepancy between the interaction generation mechanism that is employed by the MovieLens system and that of typical real-world recommendation scenarios. That is, the MovieLens dataset records $\langle user - MovieLens \rangle$ interactions, but not $\langle user - movie \rangle$ interactions. All research articles using the MovieLens dataset model the $\langle user - MovieLens \rangle$ rather than the $\langle user - movie \rangle$ interactions, making their results less generalizable to many practical recommendation scenarios in real-world settings. In summary, the MovieLens platform demonstrates an efficient and effective way of collecting user preferences to address cold-starts. However, *models that achieve excellent recommendation accuracy on the MovieLens dataset may not demonstrate superior performance in practice*, for at least two kinds of differences: (1) the differences in the contexts of user-item interaction generation, and (2) the differences in user knowledge about the item collections. While results on MovieLens can be useful as a reference, they should not be solely relied upon as the primary justification for the effectiveness of a recommendation system model.

CCS Concepts: • **Information systems** → **Recommender systems**; *Evaluation of retrieval results*;

Additional Key Words and Phrases: Recommendation evaluation, MovieLens, data analysis

ACM Reference format:
Yu-chen Fan, Yitong Ji, Jie Zhang, and Aixin Sun. 2024. Our Model Achieves Excellent Performance on MovieLens: What Does It Mean? *ACM Trans. Inf. Syst.* 42, 6, Article 159 (October 2024), 25 pages.
https://doi.org/10.1145/3675163

Authors' Contact Information: Yu-Chen Fan, Nanyang Technological University, Singapore, Singapore; e-mail: S220088@e.ntu.edu.sg; Yitong Ji, Nanyang Technological University, Singapore, Singapore; e-mail: S190004@e.ntu.edu.sg; Jie Zhang, Nanyang Technological University, Singapore, Singapore; e-mail: zhangj@ntu.edu.sg; Aixin Sun (Corresponding author), Nanyang Technological University, Singapore, Singapore; e-mail: axsun@ntu.edu.sg.

ACM 1558-2868/2024/10-ART159
https://doi.org/10.1145/3675163

1 Introduction

The rapid growth of digital platforms and data-driven services in recent years has given rise to a thriving recommendation system landscape, fueled by the continuous development and refinement of algorithms to meet the diverse needs of users. With an ever-increasing volume of data at their disposal, these algorithms leverage sophisticated techniques, such as collaborative filtering [8, 10], sequence-aware recommendation [21, 24], and deep learning [18, 30, 45], to provide users with highly personalized and relevant content.

However, alongside the undeniable benefits, concerns surrounding the proper *evaluation* of recommendation algorithms remain vital issues to address as the technology continues to advance. The evaluation of recommendation algorithms faces a multitude of challenges, primarily arising from the difficulties in reproducing results and the lack of uniform standards for comparison. For reproducibility, Ferrari Dacrema et al. [13] report that many algorithms have difficulty reproducing the results they show in their original articles, and some algorithms can be defeated by simpler algorithms. The absence of a unified set of evaluation metrics and methodologies compounds these challenges, as it hampers the ability to conduct fair and accurate comparisons between different algorithms. For example, to speed up assessment, sampled metrics evaluate algorithms based on selected negative samples. However, the results obtained from sampled metrics may be misleading, as it is inconsistent with the results of full rank [27]. Accordingly, the community has witnessed a number of efforts [42, 48] to establish standardized evaluation protocols and to ensure that advancements in the field are grounded in reliable and reproducible research. However, standardized evaluation protocols are meaningful only if *the datasets used for the evaluations are meaningful and representative*. In other words, a good understanding of each benchmark dataset used for evaluation is critical, e.g., to what extent a benchmark dataset represents a recommendation scenario, and what scenario. If a dataset well represents a practical recommendation setting, then we can expect the algorithm stands for a good chance of achieving similar performance in practice as on the dataset.

Indeed, researchers recently start to focus on an often overlooked aspect of the **recommender system (RecSys)** evaluation: *the benchmark dataset*. Specifically, the characteristics of a dataset can significantly impact the performance of various algorithms obtained on the dataset [7]. Schnabel et al. [36] explore the selection bias in implicit feedback datasets, which arises from users' selective exposure to items. Our study expands on previous studies on dataset analysis and attempts to explore a different perspective: *to what extent do we understand the user-item interaction generation mechanisms of a dataset?* The findings made along this perspective help us to better understand model performance on the dataset, and make better expectations of model performance in practical settings.

To this end, we have conducted a comprehensive and in-depth case study on the MovieLens dataset, arguably the most widely used benchmark dataset in the recommendation research community [7, 22, 26, 42]. Our analysis is conducted in three steps.

First, we conduct a literature survey, combined with our own user experiences interacting with the MovieLens platform, to gain insights into the user-item interaction generation mechanisms of the platform. The movie ratings are collected through a progressive interaction process. In particular, interactions for most users on MovieLens are collected within a very short time period (e.g., within a single day for about half of all users). The interactions from every user are much affected by the candidate movies listed on the recommended web pages at different stages during this progressive interaction collection process. The pool of candidate movies in the recommended page expands along the deepening of user interaction, thus causing differences in the recorded user interactions in different stages. Among them, the initial set of user interactions largely defines a user portrait, which affects the types of movies to be rated by this user.

Second, based on our comprehension of the user-item interaction generation mechanism, we design experiments to capture the potential impact of the data characteristics to the performance of recommendation algorithms. Specifically, we conduct ablation experiments on the raw data by masking out interactions at different stages of user interactions, and observe the performance changes of recommendation algorithms. The performance drop of the algorithm is most pronounced when we mask out the interactions that are closer to the predicted interaction. Following early studies, we also perform data shuffling on the rating sequences. The signal of causality between interactions introduced by the internal data collection mechanism of MovieLens is the main reason why sequential recommendation algorithms perform well on this dataset.

Finally, based on the findings, we discuss the discrepancy between the user-item interaction generation mechanisms of the MovieLens dataset and that of practical scenarios. Specifically, we discuss two kinds of differences. The first type of difference is the *contexts of user-item interaction generation*. On a typical practical recommendation platform, a user may consider many factors before an interaction happens, e.g., the monetary cost for purchasing a product or the time cost of watching a video. However, on MovieLens, the rating interaction is a record of what happened in the past e.g., watched a movie at somewhere on someday. That's why nearly half of the users are able to record a good number of ratings within a single day on MovieLens. The decision-making process is very different. The second type of difference is *the user knowledge about the item collections*. We would expect most users to have a reasonable understanding of different movie genres like action, comedy, and romance, as well as a few key attributes like directors and actors. The same applies to recommendation settings for books and music. However, it is unreasonable to assume that most users have a good understanding of all kinds of products available on Amazon or eBay. The information gap between users and the item collection may affect user behavior. Accordingly, models that achieve good performance on MovieLens may not show the same under a recommendation scenario where users have limited knowledge about the item collections.

In summary, our work highlights the non-negligible role of datasets for model generalization and evaluation. To the best of our knowledge, this is the first attempt to design evaluation experiments and explain algorithm performance from the perspective of the interaction generation mechanism of a dataset. Based on our findings, we argue that the context of decision makings for users when interacting with the MovieLens platform could be very different from many other kinds of decision makings when users interact with a recommender platform. As a model is trained to learn the underlying patterns from a given dataset, the recommenders that achieve excellent performance on the MovieLens dataset may not show a competitive advantage in many real-world settings. In other words, while results on MovieLens can be useful as a reference, they should not be solely relied upon as the primary justification for the effectiveness of a recommendation system model.

2 The MovieLens Interaction Generation Mechanism

In this section, we first brief the process of collecting user-movie ratings on the MovieLens platform. Based on our understanding of the data collection process, we curate a new dataset named MovieLens1518, and provide an initial analysis on the dataset.

2.1 Data Collection Mechanism

The MovieLens dataset was initially introduced by the GroupLens research team in 1998. The interactions (i.e., user ratings on movies) originate from the MovieLens recommendation platform, which aims to offer personalized movie recommendations based on users' preferences and viewing history. Over the years, the MovieLens platform has gone through a few versions to incorporate new features, improve user experience, and adapt to advancements in technology and RecSyss research. In 2015, the team published a article [17] detailing the key transformations the platform

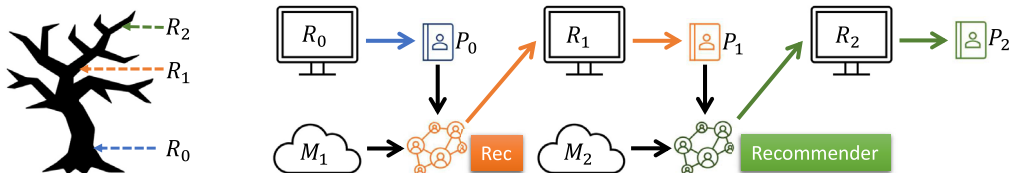


Fig. 1. Illustration of the three stages (R_0, R_1, and R_2) when a user interacts with MovieLens. From R_0 to R_2, the search space of candidate movies becomes larger and the movies recommended become more personalized, from a common root to various branches in a tree representation (the figure on the left-hand side). In the figure on the right-hand side, M_1 and M_2 represent the two pools of candidate movies to generate the list of recommended movies for users to rate, in stages R_1 and R_2, respectively. P_0 is the group-based user preference. P_1 and P_2 are the item-based preferences (i.e., movie ratings) by different internal recommendation algorithms on MovieLens with candidate pools M_1 and M_2, respectively.

underwent, which subsequently impact the resultant datasets. Specifically, the platform's intrinsic recommendation mechanism comprises two primary components: *preference elicitation* [4], and *recommendation algorithm selection* [12]. To ensure the consistency in user-item interaction collection context, hence the sampled data used in our experiments, we focus on the latest version (Version 4) of the MovieLens platform.

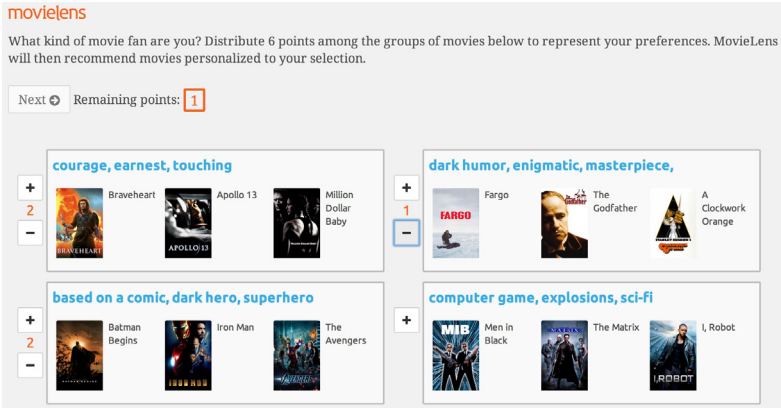
The core user feedback on MovieLens is user ratings of movies that are recommended by this platform. The platform updates a user's portrait by capturing the user's ratings on a list of candidate movies in a web page. Then, the platform generates a new list of candidate movies in the next page for the user to rate, and the process repeats. The interaction generation mechanism between a user and the platform can be divided into three main stages: *group-based preference elicitation* (R_0), *item-based preference elicitation* (R_1), and *recommendation with preferred algorithm* (R_2). Figure 1 provides an illustration of the three stages.

2.1.1 Group-Based Preference Elicitation. In the initial phase (R_0), when a new user registers on the platform, he/she is prompted to express his/her preferences on different movie groups by assigning points to each group (see Figure 2(a)). The assigned points help to establish the initial pseudo-rating profile of a user e.g., a user likes action movies more than documentary movies. Utilizing the pseudo-rating profile, the MovieLens main page displays a list of personalized recommendations from *a restricted pool of representative movies*, denoted by M_1, mostly well-known movies.

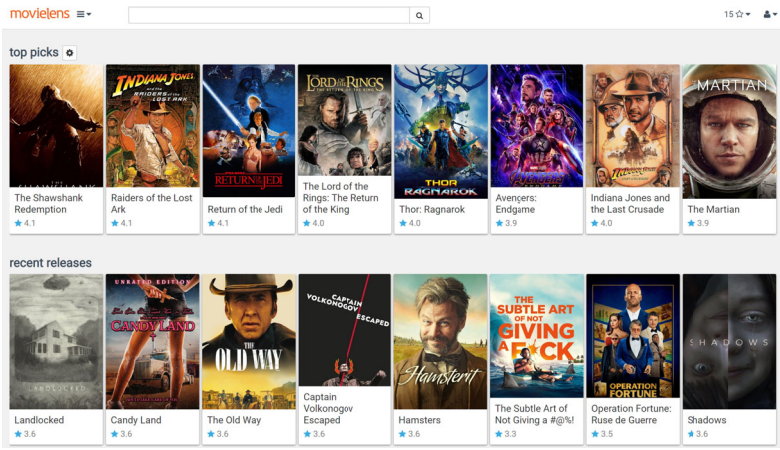
2.1.2 Item-Based Preference Elicitation. In the second stage (R_1), the platform seeks to obtain more fine-grained user preferences by prompting users to rate candidate movies from the initial personalized recommendation list (see Figure 2(b)). MovieLens requires users to surpass a certain number of ratings (i.e., 15 movies) before they can access the full-scaled system (R_2).

2.1.3 Recommendation with Preferred Algorithm. At stage R_2, MovieLens allows users to choose their preferred recommendation algorithm; consequently users can continue to rate movies among the recommended titles in an iterative manner e.g., submitting ratings of candidate movies shown on the current web page, and receiving the next list of candidate movies to rate on the next web page. The movies recommended on these pages are from M_2, a much larger pool of movies than M_1.

The platform offers four distinct algorithms: (1) a non-personalized algorithm to recommend movies with the highest average ratings; (2) the same Pick-Groups recommender in R_1; (3) an Item-Item recommender to use cosine similarity on item-mean-centered rating vectors for recommendations; (4) an SVD recommender which utilize the FunkSVD algorithm [34] to capture latent



(a) Web interface for group-based preference elicitation, *i.e.,* stage R_0. Users assign points to different movie groups. There is no individual movie ratings at this stage.



(b) Web interface for item-based preference elicitation. Users assign ratings to the recommended movies in stages R_1 and R_2. The two stages are distinguished by the different internal recommendation algorithms used on MovieLens.

Fig. 2. Screencaptures on MovieLens website for the three stages: (a) for stage R_0, (b) for stages R_1 and R_2.

patterns among the collected user-item interaction data. If a user does not choose any preferred recommendation algorithm, the default choice is Algorithm (3), i.e., an item-item recommender.

Note that, prior to Version 4, MovieLens recommended movies for users to rate strictly based on user-personalized predicted values, by its underlying collaborative filtering algorithms. With Version 4, the platform incorporates a popularity factor alongside the predicted values, in the final recommendations

$$\hat{s_{ui}} = 0.9 \cdot rank(s_{ui}) + 0.1 \cdot rank(p_i). \tag{1}$$

Specifically, the final rank score $\hat{s_{ui}}$ of a movie is a linear combination of its predicted rank score s_{ui} by a recommender and its popularity score p_i, where p_i is derived based on the number of ratings a movie received in the past year. Here, the $rank()$ function normalizes its input to $[0, 1]$ with 1 being highest rank.

Table 1. MovieLens1518 is a Curated Dataset from the MovieLens-25M Dataset, Where a User's All Interactions Were Recorded from 2015 to 2018, and Each User Has At Least 35 Ratings

Dataset	#Users	#Items	Avg #Ratings per user	Avg #Ratings per item	#Interactions
MovieLens1518	24,812	36,378	170.3	116.2	4.2M

We note that our review of the user interaction (or web interfaces) with the MovieLens platform is mainly on movie ratings. The MovieLens platform provides many other features like search for movies and add new movies [17]. As our key focus is the data collection of user-movie ratings, we do not cover these additional features offered by the MovieLens platform.

2.2 The MovieLens1518 Dataset

As aforementioned, the MovieLens platform went through different versions. Hence, the interaction mechanism between users and the platform changes accordingly, which may affect the resultant data collected. To minimize the potential impact of different versions, we extract user interaction data from the last 4 years (2015–2018) in the MovieLens-25M dataset, then remove duplicates. Note that, MovieLens-25M was released in 2019 and is currently the most recent and largest dataset ever released by GroupLens at the time of writing.[1] The reason for selecting the interactions that happened from 2015 to 2018 is that the new User Interface design and new recommendation engine (i.e., Version V4) were deployed on MovieLens in November 2014. Since then, the MovieLens platform's version consistency is maintained. As a result, from the released MovieLens-25M dataset, we curate a subset for this study and we name it the MovieLens1518 dataset, for having user-item interactions that were recorded during the 4 years from 2015 to 2018. The MovieLens1518 dataset contains about 4.2 million user-item interactions (see Table 1).

In the curated MovieLens1518 dataset, we only retain ratings from the users whose entire rating history falls within the 4 year period. This is to guarantee a comprehensive view of every user in this newly curated dataset. In addition, we remove the less active users who have fewer than 35 interactions with MovieLens, from the collected data. By setting 35 interactions as a threshold, we have good information about users' behaviors in the R_1 stage (i.e., 15 movie ratings) and also the subsequent stages (i.e., at least another 20 movie ratings). Note that, in the original MovieLens-25M dataset, each user has at least 20 ratings. In our analysis, we consider the first 15 ratings to be different from the rest, because the first 15 ratings obtained in stage R_1 are from a smaller pool of candidate movies (i.e., M_1 in Figure 1).

The 4 Yearly Subsets. In a typical RecSys evaluation, a dataset is used as a whole without considering when the user-item interactions occur; in other words, the global timeline of user-item interactions is not considered [23, 40]. As the MovieLens internal recommendation considers the popularity score of movies in the past year, we follow the same time scale and set the evaluation time window as one year and conduct independent comparative experiments year by year, from 2015 to 2018. That is, we divide the MovieLens1518 into 4 subsets according to the global timeline, each subset for ratings made in one year, from 2015 to 2018. The data partition is based on a user's last rating. That is, if a user's last rating falls in Year 2016, then all of this user's ratings belong to the 2016 subset, regardless of whether some of the earlier ratings are made in 2016 or earlier. As most users complete all their ratings in a few days, most ratings are indeed made in the indicated year. For the completeness of the experiments, we also report the results obtained on the entire 4 year data when necessary.

[1] https://grouplens.org/datasets/movielens/

3 Analysis on the MovieLens1518 Dataset

We start with the analysis on the MovieLens1518 dataset from user perspective, particularly on the amount of time users spend on the platform.

Finding 1. *On the MovieLens platform, users typically complete all their movie ratings within a very short time frame, ranging from a single day to a few days.*

We observe that 49.19%, or nearly half of all users, complete all their ratings within a single day. More than 85.6% of all users complete all their ratings within 5 days. And in many cases, a few ratings from the same user are recorded within a few seconds. In some extreme cases, the ratings are recorded with the same timestamp, making it difficult to determine the order of interactions. Specifically, in MovieLens1518, 7.53% of interactions share a same timestamp as another interaction in the same user rating sequence.

Note that, this observation is not new. Similar observations on MovieLens dataset temporal patterns are also made by Woolridge et al. [44]. In fact, the creators of the dataset also expressed concerns about the value of timestamps in their 2015 article [17]. While this observation is not new, it is worth mentioning here that MovieLens only records user ratings of movies, but not when or where the user interacts with a particular movie. In other words, the rating of a movie is based on a user's memory after he/she has watched the movie before interacting with MovieLens.

3.1 Item-Based Receptive Field Expansion

From the data collection mechanism shown in Figure 1, for a given user, the pool of candidate movies that a user can rate changes at different stages. We borrow the concept of "receptive field" to understand the MovieLens dataset. In **convolutional neural networks (CNNs)**, the receptive field of a neuron refers to the size or extent of the region in the input that affects the output of that neuron. Essentially, it's the area of the image that a neuron "sees" or processes. In our context, the receptive field is essentially the union set of items that users have interacted with at different stages of the process, acting as a proxy of the pool of recommended items.[2]

To facilitate meaningful comparisons across different user interaction stages, it's important to maintain a constant number of interactions at each stage from a fixed group of users, analogous to one layer in CNN. We fix the number of interactions to 15 at each stage. The median number of interactions per user is 98 in our dataset. Hence, we use 90 (the closest multiple of 15) as a threshold for user filtering, to ensure that we have a good number of users to be included in this study. Accordingly, users who have at least 90 interactions are used in this study. Then we derive the receptive field and compute their similarities for this fixed group of users, at different stages for *each batch of 15 ratings as one stage.*

Table 2 reports users' receptive fields in a matrix format. The reported "receptive field" is the number of unique movies that are ever rated by all users during that stage (e.g., from the 16th rating to the 30th rating for stage 16–30). The number of unique movies across all rating stages are listed in the last column of the table, for reference.

Finding 2. *Along the interaction progresses, from the initial stage (1–15) to the last stage (76–90), users are provided with a broader range of movies to rate, leading to continually expanding their receptive field.*

This observation holds on all 4 years. Further, for a given stage e.g., 1–15, the receptive field increases over time from 2015 to 2018, as more movies are released over time. We further use

[2]The optimal method for deriving the receptive field in this context is by utilizing movies listed in the impressions. However, since the MovieLens dataset lacks impressions, the rated movies are employed as a proxy.

Table 2. Distribution of Users' Receptive Fields at Different Stages

Year	#User	Receptive field (#Movies) at different stages						
		1–15	16–30	31–45	46–60	61–75	76–90	All ratings
2015	2,544	2,240	2,919	3,470	3,851	4,136	4,468	15,160
2016	3,619	2,799	3,385	3,898	4,173	4,503	4,811	20,470
2017	3,529	3,312	3,876	4,257	4,415	4,694	4,919	22,899
2018	3,599	3,690	4,088	4,491	4,692	4,937	5,250	27,596

The receptive field is the number of candidate movies derived from all users during a specific stage (e.g., the 16th to the 30th ratings indicated by 16–30) in the yearly dataset. The number of candidate movies derived from all ratings are listed in the last column.

Table 3. Intersection over Union (IoU) of Users' Receptive Fields between Consecutive Stages

Year	Intersection over Union				
	15 vs. 30	30 vs. 45	45 vs. 60	60 vs. 75	75 vs. 90
2015	0.4050	0.4735	0.4950	0.5251	0.5269
2016	0.4035	0.4610	0.5016	0.5112	0.5211
2017	0.3866	0.4456	0.4693	0.4848	0.4936
2018	0.3850	0.4296	0.4653	0.4821	0.4854

Here "15 vs. 30" refers to the IoU between the two stages "1–15" vs. "16–30"; the same naming conversion applies to the rest columns.

Intersection over Union (IoU) as the similarity metric to measure the overlap between two receptive fields, reported in Table 3.

Finding 3. *There is an increasing overlap of receptive fields across various stages of user interactions, ranging from the initial stage (1–15) to the last stage (76–90). This indicates that the divergence in user preferences is more pronounced during the initial and intermediate stages of interactions.*

The later stages of user interactions are gradually approaching the threshold of the system's candidate items. From 2015 to 2018, the receptive field increases for any given particular stage from (1–15) to (76–90). Accordingly, the IoU of receptive fields at consecutive stages drops slightly along the year dimension.

3.2 Group-Based Preference Invariance

We now focus on the possible user preference changes during the interaction.

Finding 4. *While the user's receptive field expands during the interaction process, their group-based preferences remain relatively fixed.*

Recall that in R_0, users allocate points to various movie groups. Based on these allocations, users complete the first 15 ratings in R_1. Accordingly, the genres of these 15 movies are expected to be highly focused, e.g., similar to the groups of movies assigned with higher points. Among the first 15 ratings of each user, we derive the top three genres that the user is most interested in. The result shows that nearly 90% of the first 15 rated movies belong to three genres they are most interested in, across the 4 years from 2015 to 2018 (see Table 4).

Table 4. The Ratio of Movies Falling within the Top Three Genres of Interest, among the First 15 Ratings and the Last 15 Ratings, Respectively

Year	First 15 ratings	Last 15 ratings
2015	0.8961	0.7464
2016	0.9072	0.7435
2017	0.8957	0.7322
2018	0.8912	0.7253

Further, we calculate the ratio of each user's last 15 ratings containing movies from the same three genres, to investigate potential changes in preferences after numerous additional ratings in between. Note that, the pool of movies that appear in the last 15 ratings is nearly double the size of that in the first 15 ratings. Reported in Table 4, more than 70% of the last 15 rated movies fall in the top three genres that users are initially interested in.[3]

4 Experiments

In this section, we design experiments to investigate the impact of the interaction collection process to recommender models. We first present the baseline models and evaluation metrics used in our empirical analysis. Note that, as the train/test instances change as required by the experimental setting, we detail the train/test split under each set of experiments. Codes used in our experiments are available online.[4]

4.1 Baseline and Evaluation Metric

We select seven widely used baselines from four categories: (1) memory-based methods i.e., MostPop and Item **K-Nearest Neighbors (KNN)** [10]; (2) latent factor method PureSVD [8]; (3) non-sampling deep learning method Multi-**variational autoencoders (VAE)** [30]; and (4) sequence-aware deep learning methods SASRec [25], TiSASRec [29], and Caser [43].

More specifically, *MostPop* is a non-personalized method that recommends items based solely on their popularity. In our experiments, we follow the widely adapted popularity definition, to indicate an item's popularity by the number of interactions it receives in training set, i.e., within 1 year in the 4 yearly datasets, and within 4 years for the entire dataset. *ItemKNN* is a collaborative filtering algorithm that focuses on finding similar items based on users' past interactions. It is an extension of the KNN algorithm, specifically tailored for recommendation systems. *PureSVD* is a matrix factorization technique used for recommendation systems, which decomposes a user-item interaction matrix into three smaller matrices: user factors, singular values, and item factors. *Multi-VAE* is a deep learning-based recommendation algorithm that leverages the power of VAEs to learn latent representations of users and items.

For sequence-aware models, *SASRec*, *TiSASRec*, and *Caser* are chosen as the representative models to model meaningful sequential patterns. In particular, SASRec uses a self-attention based network to model long term dependencies within user-item interaction sequence. TiSASRec further

[3]We have also conducted experiments to study the impact of the number of ratings a user has, to the ratio of the last 15 movies belonging to the top three genres. The results show that the ratios range from 67.4% to 70.7% in different years, for the top 25% users having the most number of ratings. These numbers are not far away from the 70% reported in Table 4. In other words, the number of ratings a user has does not impact user conformity to the top interested genres.
[4]https://github.com/FrankYuchen/RevistMovieLens

incorporates time interval information between user behaviors in sequence modeling process, building on top of the self-attention network. Caser, on the other hand, employs a CNN architecture to capture sequential patterns.

As for the implementation and parameter setting, non-sequential recommendation algorithms are supported by the open-source recommendation library DaisyRec 2.0 [41]. This is to ensure the reproduction of all running details and studied settings. We have verified the implementations on the MovieLens-1M dataset, and obtained consistent results using the configuration provided in the DaisyRec article [41].

Regarding SASRec and TiSASRec, we employ third-party implementations,[5] endorsed by the original authors.[6] For Caser, we use the PyTorch version provided by the authors.[7] Third-party implementations, while potentially boosting algorithm popularity, may exhibit non-negligible deviations [20] from the original author's official implementation due to nuanced adjustments in the coding. To ensure the implementations are not flawed, we provide evaluation results on the standard MovieLens-1M dataset in Table 6 in Appendix A. The results suggest that our implementations produce comparable results as those reported in their original articles.

To search for optimal hyper-parameters for all recommendation algorithms, Bayesian HyperOpt is employed to optimize the hyper-parameters concerning NDCG@10 for each model on the dataset over the course of 30 trials [13].

Evaluation Metrics. HR@k and NDCG@k are the two metrics used to evaluate the effectiveness of ranked results in our experiments. HR@k focuses solely on whether the relevant items present among the top k recommended items, while NDCG@k takes into account both the presence and ranking of the relevant items within the top-k results. As suggested by [27], we use the *full-rank* version of both metrics. That is, we rank all candidate items, instead of sampling a fixed number of negative items for ranking. The number of the top recommended items k is set to 10. We mainly discuss results measured by HR@10 as similar observations hold for results by NDCG@10.

4.2 Impact of Interaction Context at Different Stages

In view of the observations made from our earlier analysis, the first 15 ratings of a user shall be treated differently from the subsequent interactions, when exploring recommendation models on MovieLens dataset. Accordingly, we conduct an ablation experiment with the removal of the first 15 ratings of each user in the MovieLens1518 dataset. For comparison, we also report the results of removing randomly sampled 15 ratings, and the removal of the last 15 ratings from a user's training instances, respectively. For the experiments that randomly remove 15 ratings, we repeat the experiments three times with different seeds and get the average recommendation performance to reduce random error. In this set of experiments, we follow the leave-last-one-out data partition scheme. The removal of 15 ratings only applies to the training set, for all three types of removal. We keep the validation set (the penultimate interaction) and the test set (the last interaction) unchanged for a fair comparison.[8]

Finding 5. *In general, recommendation accuracy drops when* 15 *interactions are removed from the training set. The amount of drops in performance vary, depending on the types of interactions removed.*

[5] https://github.com/pmixer/SASRec.pytorch, https://github.com/pmixer/TiSASRec.pytorch
[6] The authors' GitHub pages provide links to the PyTorch implementations.
[7] https://github.com/graytowne/caser_pytorch
[8] Note that, as every user has at least 35 interactions in the MovieLens1518 dataset, removal of 15 ratings means every user has at least 20 ratings left.

According to the HR@10 scores plotted in Figure 3, the removal of 15 interactions leads to a decrease in recommendation accuracy on all baselines. It is understandable, because partial information is lost due to the removal of training data.

In general, we observe that removing the last 15 interactions leads to poorer results compared to removing the first 15 interactions, particularly for the sequence-aware models. This is because the removal of the last 15 interactions results in the loss of the most recent information in the temporal context. It is thus more difficult to predict the most recent interaction, i.e., the test instance in our leave-one-out setting. This observation applies for all baselines, except for MostPop.

One reason for the opposite observation on MostPop is that the first 15 interacted movies are from a relatively small pool of movies pre-selected by the MovieLens platform [17]. All users interact with movies in this small pool when they start to interact with MovieLens platform. Hence, the pool of movies is in an advantageous position of receiving more interactions from almost all users. MostPop, as a popularity-based method, is significantly impacted by the removal of the first 15 interacted movies, which are the popular movies.

Observe that, in Figure 3, we plot the results on the 4 yearly subsets, as well as the full dataset denoted by "4 years." In general, on the yearly subsets, all baselines show a degradation trend in performance, as the candidate movie pool becomes larger from 2015 to 2018. For most baselines, except Multi-VAE and Caser, results on the full dataset are close to the average of the 4 yearly subsets. Multi-VAE and Caser achieve the best results on the full dataset. Similar observations hold on results measured by NDCG@10 plotted in Figure 4.

In Figure 5, we plot the number of unique movies that ever appear in a user's first 15 ratings, last 15 ratings, and all ratings, by year, in the MovieLens1518 dataset. The number of movies in each category is an indication of the "receptive field" in the recommendation.[9] The significant differences in size between the pools of movies for the first 15 ratings and the last 15 ratings are consistent with the previous observation of receptive field inflation. It also corroborates the MovieLens platform operator's description of the "Item-based Preference Elicitation" stage (R_1): The recommender for new MovieLens users, which uses movie group selection at R_0 to determine which movies to recommend, only recommends from a *restricted pool* of movies.

Before we further analyze the impact of stage R_1 on recommendation system evaluation, it is necessary to revisit the problem of interaction context misalignment that existed in the previous ablation experiments. In Figure 6, we highlight the first and the last 15 movies (in different colors and patterns) rated by four example users. Note that, all the first 15 movies are rated in stage R_1, and the last 15 movies could be the results of the different number of runs of the selected recommendation algorithms. Each run of the algorithm will update what users see in the web page, and these will become candidate movies for user interaction and constitute the interaction context. Therefore, the interaction context of the test instance of each user is *different*. In fact, when a user has enough interactions with the platform, the user is able to explore more recommended candidate movies which are more personalized along his/her preference, as illustrated in the big tree on the left-hand side of Figure 1, from a common root to more personalized preferences.

The relatively fixed pool of candidate items set by the MovieLens platform for users in the R_1 stage makes the first 15 interactions of users concentrate on the most popular items (most of them are also well-known and/or classic movies). However, the context of the user's last interactions (the last 15 interactions in our case) becomes more personalized after multiple rounds of updates by the internal engine. Thus, the removal of the last 15 interactions makes it more difficult to predict the test instance correctly, especially for sequential recommendation algorithms. From experiments

[9]Note that, the numbers reported in Figure 5 are different from that in Table 2, because the latter was derived from a selected group of users (i.e., those with at least 90 ratings).

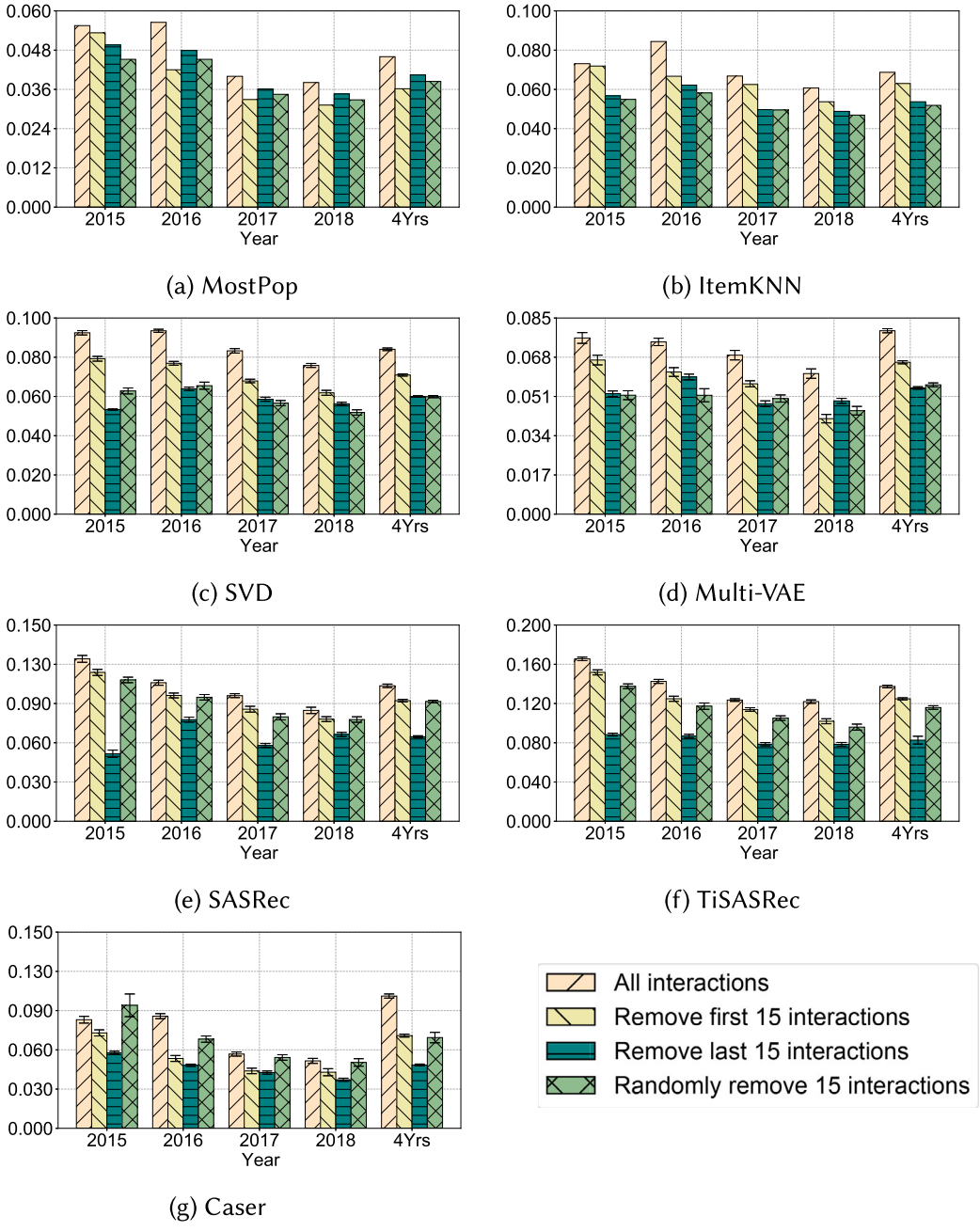


Fig. 3. The HR@10 results of removing 15 interactions from the training set of each user, on 4 yearly datasets, and the entire MovieLens1518 dataset covering all 4 years (indicated by "4 years"). We evaluate three cases: (1) removal of the first 15 interactions, (2) removal of the last 15 interactions, and (3) removal of randomly sampled 15 interactions. The 95% confidence intervals are indicated for the performance of non-deterministic algorithms.

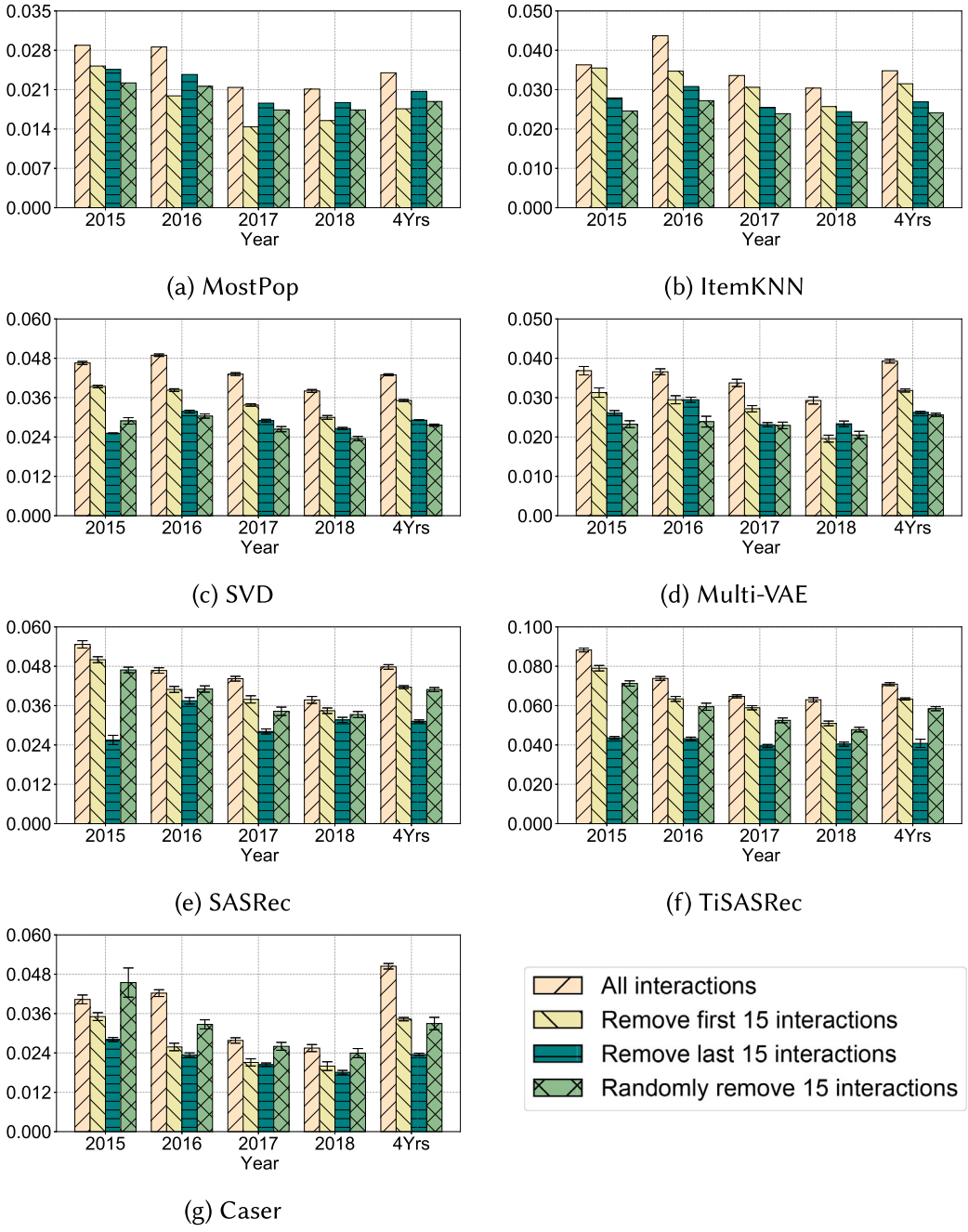


Fig. 4. The NDCG@10 results of removing 15 interactions from the training set of each user, on 4 yearly datasets, and the entire MovieLens1518 dataset covering all 4 years (indicated by "4 years"). We evaluate three cases: (1) removal of the first 15 interactions, (2) removal of the last 15 interactions, and (3) removal of randomly sampled 15 interactions. The 95% confidence intervals are indicated for the performance of non-deterministic algorithms.

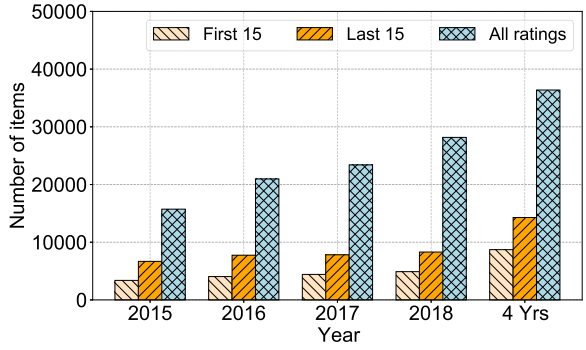


Fig. 5. Comparison of the size of the candidate movie pool at different stages, on 4 yearly subsets, and also the entire MovieLens1518 dataset covering "4 years".

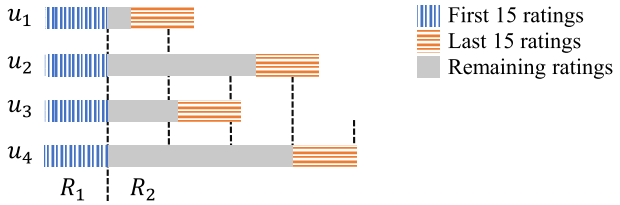


Fig. 6. An example of four users with different numbers of movie ratings. The vertical dotted line is a simple indicator of user invoking recommendation algorithms during stage R_2 i.e., the user submits her ratings on the current page and the system offers a new list of movies for rating on the next page.

with ItemKNN, SVD, and Multi-VAE, we can easily see that users' ratings of high-popularity items in the R_1 stage are of limited help to the detailed classification of users' preferences.

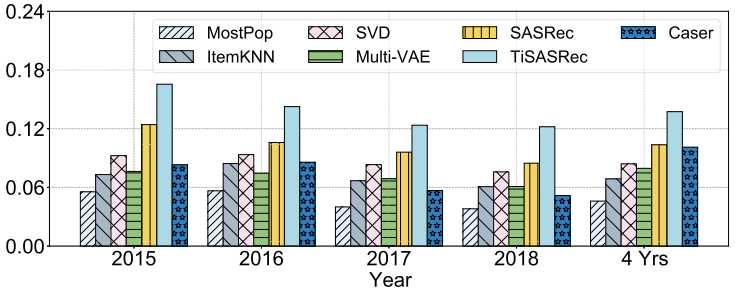
Further, we also observe the popularity dilemma posed by the first 15 interactions. When the first 15 interactions are included, the recommendation algorithm tends to treat the highly popular movies in the first 15 interactions as the correct test items, which in turn affects the recommendation performance. When the first 15 interactions are removed, the information about the user's primary interests embedded in the highly popular movies is lost, and the final result declines.

To address the popularity dilemma of MovieLens, we note that Pellegrini et al. [35] have proposed a modification: considering the co-occurrence probability of test samples with the most popular items as the objective function under popularity-sampled metrics. This inadvertently exploits group interest invariance in MovieLens: users' test items are essentially identical in group-based interest to the most popular items they rate in their first 15 interactions. Thus, the probability of co-occurrence with the most popular items allows for the determination of the preference backbone to which test items are attributed.

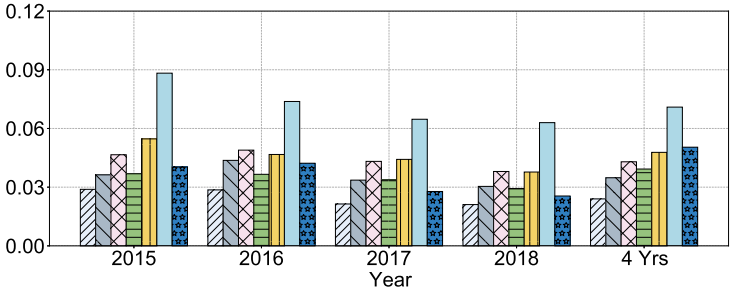
4.3 Impact of Interaction Sequence

The plots in Figure 3 do not provide a direct comparison of model performance. Figure 7 provides a direct comparison of the seven baselines without training data removal, measured by HR@10 and NDCG@10, respectively. Observe that TiSASRec is the best performer followed by SASRec, both are sequence-aware models. In general, there is a degradation trend for all models from the 2015 subset to 2018 subset.

Recall that movies rated by users in the MovieLens dataset are from a list of candidate movies recommended by an internal recommendation engine, in an iterative manner. Therefore, it is easy



(a) HR@10



(b) NDCG@10

Fig. 7. HR@10 and NDCG@10 results of seven baselines without data removal from training set, on 4 yearly datasets, and also the entire MovieLens1518 indicated by "4 years".

to see that user interactions under the MovieLens interaction mechanism are highly sequential, following a latent sequence determined by the internal recommendation engine. On the other hand, there is no significant changes in user preference based on the study on the ratio of rated movies in the top three genres, which denotes that user interaction sequences are based on a high degree of conformity to the MovieLens interaction generation mechanism.

As previously stated in Section 3, MovieLens records the user's choices and evaluations based on his or her memory among the candidate items, rather than the user's actual movie-watching behavior. Driven by the interaction generation mechanism, the exposure of a movie at different stages of user interaction is not uniformly distributed, which in turn may introduce an implicit bias in the user sequence.

To further analyze the impact of potential biases introduced into the user interaction sequence by the interaction generation mechanism, our final experiment changes the order in the original sequence by data shuffling. Again, we follow the leave-last-one-out scheme, and changes are only made on the training set for a fair comparison. Specifically, we keep the validation set and test set unchanged, get new pseudo-sequences by disrupting the order of user interaction sequences in the training set, and observe the performance changes of the sequence recommendation algorithm. We repeat the experiment three times with different seeds.

Finding 6. *After shuffling the data, the performance of sequential recommendation algorithms, which initially performed well, dropped significantly. The drop in performance of SASRec is slightly smaller, compared to TiSASRec and Caser.*

Our results (see Table 5) are partially consistent with that reported in [44], which suggests that the sequences in MovieLens are pseudo-sequences because a large part of SASRec's performance

Table 5. The Impact of Data Shuffling on the Performance of Sequential Models on MovieLens1518

Model	Metric	Original sequence	Shuffled sequence	% decrease after shuffling
SASRec	HR@10	0.1034	0.0604	41.61%
	NDCG@10	0.0464	0.0299	35.51%
TiSASRec	HR@10	0.1378	0.0722	47.61%
	NDCG@10	0.0687	0.0342	50.23%
Caser	HR@10	0.0713	0.0342	52.00%
	NDCG@10	0.0535	0.0271	49.44%

We show the average algorithm performance over 4 years.

comes from modeling the internal recommendation engine of the MovieLens system rather than true sequence information [17]. The difference is that our results are based on full-rank evaluation while the results in [44] are based on sampled evaluation. We also include two more sequence-aware models TiSASRec and Caser.

In terms of interaction contexts, data shuffling mixes candidate movies at different interaction stages. Actually, the overlap between the candidate movie pools in two adjacent interaction phases is less than 50% (see Table 3), which implies a significant difference between the different stages. Therefore, data shuffling increases the difficulty of interaction context simulation and accurate recommendations. Also, data shuffling destroys most of the potential information provided by the internal recommendation engine.

Based on all experiments so far, user interactions with MovieLens exhibit a consistent and significant pattern: the hierarchical expansion of the candidate item pool and the stability of group preference. Therefore, the efficacy of sequence-aware models on the MovieLens dataset primarily hinges on their capacity to comprehend this specific pattern. The method of sequence modeling, the various sequence stages, and the temporal distribution information collectively impact the final performance of a sequence-aware model to different degrees. The performance of the TiSASRec model is significantly enhanced by the inclusion of local temporal distribution modeling compared to SASRec, due to the more detailed modeling of the user interaction context. In addition, the performance of Caser, which is based on localized sequence information, is not stable enough on the smaller scale per-year dataset (see Figure 7).

5 Discussion

To summarize, what we have learned from the MovieLens user-item interaction generation mechanism and the MovieLens1518 dataset are the followings. (1) The movie ratings on MovieLens are collected from users through interactions at different stages. The movies for rating are recommended by an internal recommendation algorithm. Along the interactions at different stages, the receptive fields of users expand. (2) Nearly half of the users complete all their ratings in a single day and more than 85% of users complete all ratings within 5 days. (3) Users likely maintain their preferences reflected by nearly 90% of movies in the first 15 ratings fall into the top three genres of most interest, and above 70% of last 15 ratings fall into the same genres. (4) Removal of training instances (e.g., the first 15, the last 15, and randomly sampled 15) leads to decreases in recommendation accuracy, and larger drops are observed for removal of training instances closer (e.g., the last 15 ratings) to the test instances. (5) The best-performing model TiSASRec benefits from the sequential pattern of the internal recommendation, and data shuffle leads to a large drop

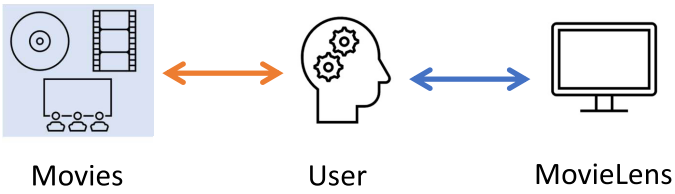


Fig. 8. Illustration of two different kinds of interactions: The $\langle user - movie \rangle$ interactions may be accumulated over a long time period through different forms of interaction e.g., movie watching in theater or on streaming platform; The $\langle user - MovieLens \rangle$ interactions typically happen within a day and user rates movies that he/she ever watched.

in performance. (6) All baseline models show a degradation trend on the yearly subsets from 2015 to 2018 due to the increase in movie candidates.

Our main focus here is how to explain these findings, and the implications of using the MovieLens dataset for model evaluation. In other words: *are similar findings expected on a typical recommendation scenario in practice?* To answer this question, we focus on the similarities and differences between the user-item interaction generation mechanisms on MovieLens and on typical recommendation platforms, from two perspectives.

5.1 The Two Types of User-Item Interactions

In our understanding, we consider that the MovieLens platform is an effective platform for *collecting* user preferences on movies (or similar items like books, music, video games, poetry, TV shows) at a large scale, *for cold-start setting*. The website is well designed to provide a small (such as 50 movies in one selection page) but personalized list of titles for a user to choose from, as a "guide" to help him/her to *recall the movies watched earlier* and then give ratings. There are two forms of user-item interactions, as illustrated in Figure 8.

— The $\langle user - movie \rangle$ interaction is between users and the individual movies which happened in the past, and is accumulative over time. A user may spend 10 years to complete the watching of Harry Potter series as these movies were released in a 10 year time frame. The decision-making to watch every Harry Potter movie could be different. In other words, a user may decide to watch a movie because of its actors and directors, simply because he/she receives a free ticket, or by following the recommendation on a platform, among many possible reasons. For each of such decisions, there could be consideration of the monetary and time cost of watching a movie.
— The $\langle user - MovieLens \rangle$ interaction between the users and the MovieLens platform. Following the "guidance" of the MovieLens' internal recommendation, the user recalls the movies that he/she had watched before and records his/her ratings.

The two types of interactions are completely different. In particular, when interacting with the MovieLens platform, a user has watched a good number of movies. It is not difficult to *summarize* the kinds of movies (e.g., the top three genres) he/she likes. Hence, this rating collection process can be completed within a short time, and the process of recalling the movie titles is likely from the small pool of more famous movies (in stage R_1) to a much larger pool of movies of similar kinds (in stage R_2). There are also no preference changes over time as users usually complete all ratings within a short time based on their developed preferences on movies over the time before the interaction with MovieLens. More importantly, the sequential pattern among the ratings from a user does not necessarily reflect the actual sequence of a user watching these

movies [17, 22]. Rather, the sequence is the result of the guidance through the rating and recommendation iterative process on MovieLens. Even if there were personal preference changes when the user watched these movies along time, such changes cannot be reflected in the rating sequence.

As a user has already watched a movie, there is no decision-making process on whether a new movie shall be watched with either monetary, time or other forms of cost. However, for a typical online recommender platform, a user often needs to make a judgment before interacting with a "new" item being recommended, among the choices available at that decision time. In other words, in a typical recommendation platform, we expect the model to recommend "new" items of the user's interest that the user has not interacted with before. The MovieLens dataset does not reflect such a setting. Again, we are not the first to question the usefulness of the MovieLens datasets from this perspective. In 2012, Jannach et al. [22] discussed that "the context of movie consumption (Am I watching a movie alone or with friends? Am I looking for entertainment or for intellectual challenge?) is largely not taken into account in the majority of articles" and the prediction on the dataset could be "accurate but not valuable." Nevertheless, to the best of our knowledge, our article is the first attempt to formalize the two kinds of user-item interactions i.e., the $\langle user - movie \rangle$ interactions and the $\langle user - MovieLens \rangle$ interactions, for a clearer comprehension of the dataset. In short, all research articles using the MovieLens dataset model the $\langle user - MovieLens \rangle$ rather than the $\langle user - movie \rangle$ interactions, making their results less generalizable to many practical recommendation scenarios in real-world settings.

Yu and Sun [46] argue that "a model is trained to answer the same information need in a similar context (e.g., the information available), for which the training dataset is created." That is, a model learns user behavior from the training data, with the assumption that similar user behavior will happen when the model is applied in a different or real-world setting. In our discussion, we consider that the contexts for the $\langle user - MovieLens \rangle$ interactions and the $\langle user - movie \rangle$ interactions to be very different. Hence, a model that achieves excellent performance on MovieLens may not show superior performance in many practical recommendation scenarios, because there are many more factors that would affect the user's behavior in each decision making, but the decision-making for the $\langle user - MovieLens \rangle$ interaction is at nearly no cost.

Evaluating RecSys models is challenging, particularly under an offline setting, because many factors that may affect user online behavior are not well captured in an offline dataset. Gunawardana et al. [16] state that "the goal of the offline experiments is to filter out inappropriate approaches, leaving a relatively small set of candidate algorithms to be tested" online, and "it is necessary to simulate the online process where the system makes predictions or recommendations." As illustrated in Figure 8, the kind of $\langle user - MovieLens \rangle$ interactions collected in the MovieLens dataset cannot be used to simulate an online process for users to make judgements on what movies to watch next i.e., $\langle user - movie \rangle$ interactions. Although the recommended usage of the MovieLens dataset is for algorithm comparison in research articles [17], we hope that through our analysis, researchers have a better understanding of the results obtained on the MovieLens dataset.

5.2 The Information Mismatch between User and Item Collection

In MovieLens, both users and the system have a clear understanding of the collection of items, i.e., the movies. A typical user would have a basic comprehension of how to characterize a movie with its actors, directors, genre and so on. The same applies to music, books, hotels and flights. However, in many other recommendation scenarios, users may not have a clear understanding of the organization and distribution of all items, e.g., all products available for purchase on e-commerce sites, and the different kinds of advertisements to be recommended to users. Under such

settings, users may not know the existence of certain types of items and do not have a full picture of all possible types of items that are available for recommendation. The degree of information mismatch between the user and the item collection of the MovieLens dataset and that of other recommendation scenarios could be very different. Hence, a model that is good at learning certain patterns from the MovieLens dataset may not be good at learning patterns from datasets obtained from other platforms.

In short, there are two perspectives to understanding the MovieLens data. First, the MovieLens dataset does capture user preferences on movies e.g., the types of movies a user prefers to watch, from all movies that a user has watched so far. Second, the MovieLens dataset is more of a collection of the *results* of user-item interactions, where users have good knowledge about the item properties. Hence, the model performance made on the MovieLens dataset could be a good reflection of to what extent the model matches the underlying recommendation algorithm well (e.g., item-item similarity used to power MovieLens); the performance may not be a good reflection of what the model would achieve in real-world settings where a user need to make a judgment when facing newly recommended items of little knowledge, and the user has never interacted with these items before, and there could be cost incurred to interact with these items.

5.3 Information Collection for Cold Starts

Cold start is one of the major challenges in many RecSyss, where the system does not know much about the new users (i.e., user cold start) and/or the new items (i.e., item cold start). Preference elicitation in the cold-start phase determines the quality of subsequent recommendations and thus has been studied extensively. Researchers strive to propose new preference elicitation methods to better address the cold-start problem. For instance, Sepliarskaia et al. [37] define preference elicitation as an optimization problem to generate *static preference questionnaires* at a lower cost. Graus and Willemsen [15] argue that the choice-based interface requires less user effort than rating-based interfaces, and leads to a more satisfying recommendation experience.

The MovieLens platform in this perspective demonstrates an efficient and effective way of collecting user preferences in a personalized iterative process. In particular, it starts with a group-based preference elicitation, and then guides the users to "recall" the movies that they have interacted with through an recommendation process. As the result, for a new user, the platform is able to collect a good number of movie ratings within a very short time (e.g., within a day for nearly half of all users), and with very little effort from users. These ratings collected well reflect the user preferences, which is demonstrated by the conformity to the top three genres, serving an effective way to understand users. In this sense, the MovieLens dataset can be considered as the result of static preference questionnaire collected through a well designed functional web interface. We use the term "static preference" to describe the user preference collected in MovieLens, for the reason that the user preference here is inferred from all ratings submitted within a short period. In other words, the user preference is a summary of all the movies he/she has watched over a much longer time period in no particular order. As there is no time point when a user watches which movie, no preference change over time can be inferred.

5.4 Is It a Good Idea to Evaluate RecSys Models on MovieLens?

The succinct answer is no. Our analysis indicates that the user-item interactions recorded in the MovieLens dataset represent engagements between users and the MovieLens platform, where users are guided to recall movies they have previously watched. We perceive this setup as significantly divergent from typical recommendation scenarios encountered in practice. Notably, the MovieLens

dataset emerges as a distinct outlier in the distribution of certain statistical features, as previously observed in studies [7, 43]. We believe that performances obtained on the MovieLens dataset do not adequately reflect a model's expected performance in real-world scenarios. Therefore, results obtained solely from the MovieLens dataset cannot be relied upon as indicative of a model's online performance post-deployment.

On the other hand, MovieLens stands out as one of the most popular datasets in the field of RecSyss [7, 41]. Results derived from MovieLens serve as valuable references for researchers to validate their implementations. For instance, researchers can compare the outcomes of their own implementations with the results reported by the authors on the MovieLens dataset, as in Appendix A, Table 6.

In short, while providing results on MovieLens for reference purposes is beneficial, it should not serve as a strong justification for the effectiveness of a proposed model. Models should be evaluated on a variety of datasets, not relying solely on the MovieLens dataset.

6 Related Work

Our research focuses on the analysis of the MovieLens dataset with an aim to identify its implications on recommender evaluations. Accordingly, we review the related studies on RecSys dataset analysis, the study of biases in RecSys, and the common issues in RecSys evaluation.

6.1 RecSys Dataset Analysis

Existing dataset related studies in RecSyss have placed their focuses on dataset analysis. Deldjoo et al. [9] emphasize that the user-item rating matrix structure and the rating distribution largely affect the robustness of the collaborative filtering model in the case of shilling attacks. Luca [32] examine the causal impact of Yelp consumer ratings on restaurant demand with a regression discontinuity approach. The authors then test whether consumers use Yelp reviews in a way that is consistent with standard Bayesian learning models. The results suggest that consumers show selectivity in using reviews, being more responsive to visible quality changes and ratings with more information. Leino and Räihä [28] explore user behaviors in the Amazon online store using on-location interviewing and observation. They are especially interested in what kind of strategies users had developed for utilizing algorithm-based recommendations and customer reviews to discover items of interest. Steck et al. [39] discuss the challenges and lessons learned in using deep learning for RecSyss at Netflix. It highlights that deep learning models show significant improvements only when combined with additional heterogeneous features like timestamps and by addressing issues of offline-online metric alignment. In [44], the authors demonstrate that the interaction history in the MovieLens dataset is pseudo-sequential because the actual order of these interactions is unknowable. Similar to our study, these articles mostly focus on one specific dataset. Considering the arbitrary use of various datasets, the dataset dilemma in RecSyss has been raised in [7]. The authors cluster datasets from different domains based on structural and distributional characteristics, and examine the performance rankings of algorithms across different clusters.

6.2 Biases in RecSyss

The recommendation system can be abstracted as a feedback loop among three key components: User, Data, and Model [6]. Biases occur in different stages including data collection (User to Data), model learning (Data to Model), and final recommendation (Model to User). Our study pays more attention to biases in collected data and recommendation results. Specifically, Chen et al. [5] formally present four common pitfalls in training and evaluating recommendation algorithms. Among them,

two pitfalls are highly relevant to this study: (1) popularity bias: trained models could be biased toward highly reachable products because these items are more likely to be treated as positive training instances. (2) exposure bias: interactions collected from an online platform are influenced by its deployed RecSys. Hence, the distribution of interactions is fundamentally different from the interactions with no exposure to a RecSys.

The long-tail distribution of user interactions is frequently observed in recommendation system data, where a small portion of highly popular items tends to dominate user interactions. When trained on such data, models often have the propensity to recommend popular items over less unpopular ones. MovieLens datasets, as a typical long-tail dataset, is often used as an experimental dataset for popularity bias and debiasing. Abdollahpouri et al. [1] have empirically verified the impacts of popularity bias on different stakeholders, such as users and suppliers. Further, they propose metrics to measure the average deviation of the recommendations in terms of item popularity. Steck et al. [39] investigate the effects of four factors, namely inherent audience size imbalance, model bias, position bias, and closed feedback loop, on popularity bias in RecSyss by simulating dynamic recommendation experiments. They conclude that audience size imbalance and model bias are the main factors contributing to popularity bias. Apart from this, a dynamic debiasing strategy and a novel False Positive Correction method are proposed to remove popularity bias in dynamic scenarios. Exposure bias arises when users have only seen a subset of specific items and unobserved interactions are not necessarily dislikes. Liu et al. [31] point out that exposure is affected by the policy of intrinsic recommendation engines, which determine which items to show to users. Schnabel et al. [36] emphasize that users' active search behavior is also a factor of exposure and introduce a debiasing method employing inverse propensity weighting to correct bias in observed data.

Our analysis in this article offers another perspective to understand bias in the MovieLens dataset. The ratings on MovieLens are naturally biased due to two reasons. First, all users start with the 15 ratings from a small pool of representative movies, and the number of ratings per user varies significantly. Second, as the ratings are based on user memory of the movies that they have watched before, more ratings are expected to more successful movies.

6.3 Evaluation Issues in RecSyss

In the burgeoning field of RecSyss, evaluation has been an important research topic. In [13] and [11], the authors highlight the importance of establishing benchmark evaluation to avoid unfair comparisons and to ensure the reproducibility of recommendation algorithms. With that in consideration, multiple toolkits have been released to benchmark the recommendation tasks, and facilitate the development and evaluation of recommendation models [2, 14, 41, 49].

Apart from the development of toolkits, recent years have witnessed an increasing number of research articles discussing offline evaluation options for RecSyss. Sun et al. [42] categorize the offline evaluation options into eight steps, including dataset selection, dataset filtering techniques, baseline model selection, objective function selection, negative sampling method selection, dataset splitting options, evaluation metric options and hyperparameter tuning techniques. Among these options, *dataset splitting options* have garnered significant research interests. In [3, 33], the authors demonstrate the varying outcomes due to different data splitting strategies. They argue that varying outcomes could lead to misinterpretation of results. Ji et al. [23] further explain the varying outcomes from a global timeline perspective. The key finding is, the data splitting strategies that ignore the global timeline, e.g., leave-one-out split, suffer from the data leakage issue, thus leading to unpredictable recommendation performance. Other studies focus on the options of *evaluation metric*: accuracy metric and beyond-accuracy metric [19, 38, 47]. Furthermore, the authors in

[27, 48] delve into the research of sampled metric, and show the potential bias induced when using incomplete candidate list in evaluation.

Although *dataset selection* has been listed as an important step of offline evaluation in [42], it does not receive much research attention compared to *data splitting strategies* and *evaluation metrics*. In many studies [42, 48], it is suggested that researchers should select research datasets considering their popularity (whether they are frequently used by recent academic articles) as well as domain diversity. There is a lack of study that delves into the collection and construction process of a dataset to facilitate the dataset selection in offline evaluation. In this work, we zoom in on a particular RecSys dataset: MovieLens. Specifically, we understand the MovieLens dataset from its collection process.

7 Conclusion

In this study, we conduct a thorough analysis of the extremely popular MovieLens dataset in the field of RecSyss from the perspective of interaction generation mechanisms. We demonstrate the interaction generation mechanism of the latest version of the MovieLens dataset, including the detailed user interaction process and the built-in recommendation algorithm at different stages. In addition, we designed targeted experiments based on the interactive generation mechanism to observe whether the special data characteristics caused by the interaction generation mechanism of the MovieLens dataset actually lead to different observations or conclusions drawn from empirical studies. Our results demonstrate that interaction generation mechanisms can have a significant impact on data characteristics, which in turn can cause profound and unpredictable perturbations to experimental results.

The recommendation generation mechanism of the MovieLens platform notably impacts the data dynamics of user-platform interactions, thereby facilitating certain models to achieve superior performance as anticipated. There is no doubt that the MovieLens platform demonstrates the best way to collect user preferences with minimal effort from users and thus address cold-start problems. On the other hand, we argue that the recommendation models that achieve excellent performance on the MovieLens dataset may not show the same in reality by contrasting the interaction generation mechanism in MovieLens and other practical recommendation scenarios. Results on the MovieLens dataset hence should not serve as a strong justification for the effectiveness of a proposed model.

Our analysis is limited to the MovieLens dataset, one of the most widely used datasets in RecSys research. While not all findings reported in this article are new, we hope to provide a comprehensive understanding of the dataset from the perspective of user-item generation mechanism. The findings will be helpful to researchers, particularly the young researchers who are new to RecSys, to better interpret the results of their models on benchmark datasets. While the methodology used to analyze the MovieLens dataset may not be applicable to all other RecSys datasets, we strongly encourage researchers to review the mechanisms by which the user-item interactions were generated when using datasets as benchmarks for recommenders beyond their initial scope.

Appendix

A Additional Results

Table 6 lists two sets of results of SASRec, TiSASRec, and Caser, on the standard MovieLens-1M dataset. Comparing the results obtained in our experiments with those reported in the original articles, we find that the implementations used in our experiments produce comparable results to those reported in the original articles.

Table 6. Results on the Standard MovieLens-1M Dataset, Reported in Their Original Articles, and Obtained by Using Our Implementations, Respectively

Model	Metric	Article	Experiment
SASRec	HR@10	0.8245	0.8238
	NDCG@10	0.5905	0.5964
TiSASRec	HR@10	0.8038	0.8041
	NDCG@10	0.5706	0.5697
Caser	Prec@1	0.2502	0.2894
	Recall@1	0.0148	0.0181
	mAP	0.1507	0.1741

References

[1] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2020. The connection between popularity bias, calibration, and fairness in recommendation. In *Proceedings of the ACM Conference on Recommender Systems*. 726–731.

[2] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. 2021. Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 2405–2414. DOI: https://doi.org/10.1145/3404835.3463245

[3] Rocío Casimnamares, Pablo Castells, and Alistair Moffat. 2020. Offline evaluation options for recommender systems. *Information Retrieval Journal* 23, 4 (2020), 387–410.

[4] Shuo Chang, F Maxwell Harper, and Loren Terveen. 2015. Using groups of items for preference elicitation in recommender systems. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work & Social Computing*. 1258–1269.

[5] Hung-Hsuan Chen, Chu-An Chung, Hsin-Chien Huang, and Wen Tsui. 2017. Common pitfalls in training and evaluating recommender systems. *ACM SIGKDD Explorations Newsletter* 19, 1 (2017), 37–45.

[6] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

[7] Jin Yao Chin, Yile Chen, and Gao Cong. 2022. The datasets dilemma: How much do we really know about recommendation datasets? In *Proceedings of the ACM International Conference on Web Search and Data Mining*. 141–149.

[8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the ACM Conference on Recommender Systems*. Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker (Eds.), ACM, New York, NY, 39–46. DOI: https://doi.org/10.1145/1864708.1864721

[9] Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, and Felice Antonio Merra. 2020. How dataset characteristics affect the robustness of collaborative recommendation models. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 951–960.

[10] Mukund Deshpande and George Karypis. 2004. Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177. DOI: https://doi.org/10.1145/963770.963776

[11] Yushun Dong, Jundong Li, and Tobias Schnabel. 2023. when newer is not better: Does deep learning really benefit recommendation from implicit feedback? In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. ACM, New York, NY, 942–952. DOI: https://doi.org/10.1145/3539618.3591785

[12] Michael D. Ekstrand, Daniel Kluver, F. Maxwell Harper, and Joseph A. Konstan. 2015. Letting users choose recommender algorithms: An experimental study. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 11–18.

[13] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the ACM Conference on Recommender Systems*. ACM, New York, NY, 101–109.

[14] Scott Graham, Jun-Ki Min, and Tao Wu. 2019. Microsoft recommenders: Tools to accelerate developing recommender systems. In *Proceedings of the ACM Conference on Recommender Systems*. ACM, New York, NY, 542–543. DOI: https://doi.org/10.1145/3298689.3346967

[15] Mark P. Graus and Martijn C. Willemsen. 2015. Improving the user experience during cold start through choice-based preference elicitation. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 273–276.

[16] Asela Gunawardana, Guy Shani, and Sivan Yogev. 2022. Evaluating Recommender Systems. In *Recommender Systems Handbook* (3rd ed.). Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.), Springer US, New York, NY, 547–601. DOI: https://doi.org/10.1007/978-1-0716-2197-4_15

[17] F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 1–19.

[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, 173–182.

[19] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems* 22, 1 (2004), 5–53. DOI: https://doi.org/10.1145/963770.963772

[20] Balázs Hidasi and Ádám Tibor Czapp. 2023. The effect of third party implementations on reproducibility. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23)*. Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song (Eds.), ACM, New York, NY, 272–282. DOI: https://doi.org/10.1145/3604915.3609487

[21] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations*. Yoshua Bengio and Yann LeCun (Eds.).

[22] Dietmar Jannach, Markus Zanker, Mouzhi Ge, and Marian Gröning. 2012. Recommender systems in computer science and information systems - A landscape of research. In *Proceedings of the E-Commerce and Web Technologies (EC-Web)*, Lecture Notes in Business Information Processing, Vol. 123, Springer, 76–87. DOI: https://doi.org/10.1007/978-3-642-32273-0_7

[23] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2023. A critical study on data leakage in recommender system offline evaluation. *ACM Transactions on Information Systems* 41, 3 (2023), 1–27.

[24] Wang-Cheng Kang and Julian J. McAuley. 2018a. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 197–206. DOI: https://doi.org/10.1109/ICDM.2018.00035

[25] Wang-Cheng Kang and Julian McAuley. 2018b. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[26] Anastasiia Klimashevskaia, Dietmar Jannach, Mehdi Elahi, and Christoph Trattner. 2024. A survey on popularity bias in recommender systems. *User Modeling and User-Adapted Interaction*. DOI: https://doi.org/10.1007/s11257-024-09406-0

[27] Walid Krichene and Steffen Rendle. 2022. On sampled metrics for item recommendation. *Communications of the ACM* 65, 7 (2022), 75–83.

[28] Juha Leino and Kari-Jouko Räihä. 2007. Case Amazon: Ratings and reviews as part of recommendations. In *Proceedings of the ACM Conference on Recommender Systems*. 137–140.

[29] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 322–330.

[30] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the World Wide Web Conference*. 689–698.

[31] Dugang Liu, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2020. A general knowledge distillation framework for counterfactual recommendation via uniform data. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 831–840.

[32] Michael Luca. 2016. Reviews, reputation, and revenue: The case of Yelp.com. In *Com (March 15, 2016). Harvard Business School NOM Unit Working Paper 12–016*.

[33] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2020. Exploring data splitting strategies for the evaluation of recommendation models. In *Proceedings of the ACM Conference on Recommender Systems*. 681–686.

[34] Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, Vol. 2007. 5–8.

[35] Roberto Pellegrini, Wenjie Zhao, and Iain Murray. 2022. Don't recommend the obvious: estimate probability ratios. In *Proceedings of the ACM Conference on Recommender Systems*. 188–197.

[36] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1670–1679.

[37] Anna Sepliarskaia, Julia Kiseleva, Filip Radlinski, and Maarten de Rijke. 2018. Preference elicitation as an optimization problem. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 172–180.

[38] Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, and Shaoping Ma. 2019. How good your recommender system is? A survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics* 10, 5 (2019), 813–831. DOI: https://doi.org/10.1007/s13042-017-0762-9

[39] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. 2021. Deep learning for recommender systems: A Netflix case study. *AI Magazine* 42, 3 (2021), 7–18.

[40] Aixin Sun. 2023. Take a fresh look at recommender systems from an evaluation standpoint. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. ACM, New York, NY, 2629–2638. DOI: https://doi.org/10.1145/3539618.3591931

[41] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. 2022. DaisyRec 2.0: Benchmarking recommendation for rigorous evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 7 (2022), 8206–8226.

[42] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *Proceedings of the ACM Conference on Recommender Systems*. 23–32.

[43] 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.

[44] Daniel Woolridge, Sean Wilner, and Madeleine Glick. 2021. Sequence or pseudo-sequence? An analysis of sequential recommendation datasets. In *Proceedings of the Perspectives@ RecSys*.

[45] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ACM International Conference on Web Search And Data Mining*. 153–162.

[46] Mengying Yu and Aixin Sun. 2023. Dataset versus reality: Understanding model performance from the perspective of information need. *Journal of the Association for Information Science and Technology* 74, 11 (Oct. 2023), 1293–1306. DOI: https://doi.org/10.1002/asi.24825

[47] Eva Zangerle and Christine Bauer. 2023. Evaluating recommender systems: survey and framework. *ACM Computing Surveys* 55, 8 (2023), 170:1–170:38. DOI: https://doi.org/10.1145/3556536

[48] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. 2022. A revisiting study of appropriate offline evaluation for top-N recommendation algorithms. *ACM Transactions on Information Systems* 41, 2 (2022), 1–41.

[49] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the ACM International Conference on Information & Knowledge Management*. 4653–4664.

Received 31 July 2023; revised 14 March 2024; accepted 8 June 2024