Sequence or Pseudo-Sequence?

An Analysis of Sequential Recommendation Datasets

Daniel Woolridge[1], Sean Wilner[1] and Madeleine Glick[1]

[1]*Vody LLC. Los Angeles, CA, USA*

Abstract

Sequential recommendation aims to model a user's preferences by looking at the order of interactions in a user's history. The evaluation of such algorithms requires robust datasets with genuine sequential information. In this work we analyze the timestamp information of several commonly used datasets and show that reported timestamps are not indicative of meaningful sequential order. In the datasets explored, significant numbers of users have interactions occurring at identical timestamps. The actual order of these interactions is therefore unknowable; the interaction history is pseudo-sequential. We find that randomly shuffling the order of interactions has minimal impact on the performance of a leading sequential recommender. Particular attention is paid to MovieLens because of its frequency of use in the field of sequential recommendation. Our findings motivate the necessity for new datasets with more meaningful ordering for the evaluation of sequential recommenders.

Keywords

Datasets, Recommendation, Sequential Recommendation, MovieLens, CEUR-WS

1. Introduction

The ubiquity of and necessity for recommendation systems has given rise to an explosion of research in the field. Recommendation algorithms are studied and implemented for use in domains spanning media, e-commerce, social networks and more. The landscape of recommendation research consists of a plethora of techniques, most of which attempt to model the interactions between users and items in order to predict the items with which a user is most likely to interact. Sequential recommendation, an increasingly popular trend in the field, works by taking into account not only the users' interaction history but the *order* of those interactions as well. The goal of a sequential recommender is to utilize and exploit sequential patterns in historical user behavior [1, 2, 3, 4].

As with any burgeoning field, the ability to accurately benchmark and compare results across various datasets, models, and metrics is essential. Benchmarking the relative performance of various recommendation algorithms on publicly available datasets is a core part of the research and development of such systems. It is therefore of utmost importance to ensure the validity of various benchmark datasets for accurately conducting research. Throughout the machine

Perspectives on the Evaluation of Recommender Systems Workshop (PERSPECTIVES 2021), September 25th, 2021, co-located with the 15th ACM Conference on Recommender Systems, Amsterdam, The Netherlands

dan@vody.com (D. Woolridge); sean@vody.com (S. Wilner); madeliene@vody.com (M. Glick)

https://www.vody.com (D. Woolridge)

0000-0003-2415-5528 (D. Woolridge); 0000-0002-8382-3488 (S. Wilner); 0000-0003-3042-2039 (M. Glick)

learning community emphasis is being increasingly placed on verifying all aspects of the datasets used to benchmark models. Recent work on biases in image datasets [5, 6], language datasets [7, 8], and the models trained on them focus primarily on the ethical and societal impacts of these biases and how their presence and lack of mitigation can affect the wider world. Others have looked at the frequency of label errors in various commonly used datasets spanning multiple domains [9]. Misunderstanding and misuse of input data can lead to erroneous conclusions that become tainted seeds for subsequent works. One such example, brought to attention by Li et al. [10], shows how experimental procedure can imbue datasets with non-signal correlations that models will pick up on and utilize. In an analogous manner, our work analyzes time-sequence data in various recommendation datasets and calls into question their validity for assessing the performance of sequential recommendation algorithms.

There are two main approaches to assessing the performance of a recommendation system, online and offline tests. In the case of online testing, researchers are able to assess the performance of two or more systems by using those systems to provide recommendation for different user segments [11]. Often, researchers do not have access to online tests for recommendation and thus have to rely on offline tests [12]. A core assumption for offline tests is that the dataset on which they are conducted accurately reflects a real-world recommendation scenario, and if not, the differences are established and understood.

In this work, we begin by exploring the timestamp information in several popular datasets used for evaluating sequential recommendation algorithms. We find that for some of these datasets the mere presence of timestamp information is not indicative of task relevant sequential information. We discuss the construction of the datasets and explore issues therein. Additionally, we show the impact of these issues by conducting experiments with a popular sequential recommendation algorithm. Our findings call into question the validity of using these datasets for the evaluation of sequential recommenders.

2. Related Work

2.1. Sequential Recommendation

The task of sequential recommendation is to provide relevant items to users by using the users' historical interaction data and exploiting patterns between subsequent items. There has been much work done in this area which we briefly summarize below[1].

2.1.1. Baselines and Earlier Models

Simple baselines are often used to compare against more sophisticated methods. One frequently used baseline is to rank items by their popularity and provide these as recommendations for each user. Another simple yet performant baseline is to use ItemKNN, a K-Nearest Neighbors approach on the items [15].

One particularly successful class of approaches has been to explore historical sequences using K-th order Markov models to model stochastic transitions between items by utilizing sequential patterns [16]. Rendle et al. [3] have combined Markov Chains (MC) with Matrix Factorization

[1]We refer to Quadrana et al. [13] and Campos et al. [14] for deeper studies on the subject.

in their work Factorized Personalized Markov Chains which, although promising, struggles to deal with sparsity issues. As an attempt to address this problems He and McAuley [4] introduce Fossil, a model that fuses item similarity models with Markov Chain models.

2.1.2. Recurrent Models

Another class of approaches use Recurrent Neural Networks (RNNs) and their extensions to tackle the problem of sequential recommendation. Hidasi et al. [17] use an RNN architecture for session recommendation. Quadrana et al. [18]. extend this work by incorporating user interests via Gated Recurrent Unit (GRU) layer across user sessions. Zhu et al. [19] use a time interval aware Long Short-Term Memory (LSTM) model in an attempt to better capture both long and short term interactions in a users history. A key drawback of these models is that they require large amounts of dense data to perform well.

2.1.3. Attentional Models

With the success of attentional methods and transformers in multiple domains with sequential information, it is only natural that they would be applied to the problem of recommendation. Kang and McAuley [1] present SASRec, a self-attention model that is able to capture both long term semantics and provide recommendations based on a few salient actions. The success of this model has prompted many children and extensions. One such extension, BERT4Rec, has been developed by Sun et al. [2]. Inspired by the popular language model BERT, BERT4Rec uses bi-directional self attention to better model users' behavior sequences and deal with potentially noisy input sequences [20]. Ying et al. [21] introduce a 2-level hierarchical attention network in an attempt to better capture long and short term interests of users.

2.2. Evaluation and Benchmarking

The lack of effective benchmarks for evaluation of recommendation algorithms is a critical issue facing the community. Although there have been recent dedicated works dissecting vision, language and audio datasets [9] and the effect of errors therein on benchmark validity, the field of recommendation still lags behind in this area. Conference tracks are being dedicated to datasets and benchmarks[2][3], and efforts are being made to properly review and badge artifacts [22]. Various attempts to standardize dataset versioning have been proposed [23, 24, 25] but have yet to be widely adopted.

Evaluating recommendation algorithms is difficult [26], and despite many attempts to standardize frameworks [27, 28, 29, 30, 31] the field as a whole still lacks the consistency desired. Said and Bellogín [32] explore four aspects of recommendation contributions pertaining to their reproducibility, the dataset, the evaluation framework, data details, and algorithmic details. Within this structure, our work focuses primarily on the dataset and data details aspects.

A recent and vitally important publication by Rendle et al. [26] shows how several baselines can be tuned to outperform reported results, thereby calling into question many results from the

[2]https://neurips.cc/Conferences/2021/CallForDatasetsBenchmarks
[3]https://recsys.acm.org/recsys21/perspectives/

previous years. Sun et al. [29] provides an extensive look at many prominent recommendation contributions. As shown in their Figure 1B, MovieLens-1M is the most frequently used dataset for evaluating recommendation algorithms, appearing in just over 30% of the 85 papers studied. The other MovieLens datasets explored (100K, 10M, and 20M) appear less frequently but all are in the top 15 datasets by popularity. In Figure 5A of [29], results are shown on baseline models using the MovieLens dataset split randomly or split by timestamp. The authors claim that the time-aware split better simulates the real recommendation scenario, which may be true *if* the timestamps represent a realistic interaction sequence.

Gruson et al. [12] discuss the challenges of offline recommendation evaluation and specifically point out that some datasets include biases introduced in their construction, whether through the user interface, internal recommendation algorithm or otherwise. Ji et al. [33] raise concerns with offline evaluations on datasets which ignore the global timeline of interaction sequences in the data. When datasets are collected over many years some items may not be available for the entire duration of the data collection, thereby introducing biases that should be accounted for in evaluation. In a similar vein, our work takes a deeper look at the local timestamp information present in some leading recommendation datasets.

3. Datasets

Offline evaluation of recommendation systems requires robust datasets that are applicable to the task under consideration. For sequential recommendation this means that datasets should contain genuine order information that serves as a close proxy to the real world scenario being simulated. Our primary focus in this work is on the timestamp information that most recommendation datasets include and is often used to infer the order of interactions.

The timestamps for the datasets we explore are provided either as a date format or in Unix time. When presented Unix time, meaningful relations are obscured as the scale between timestamps is less apparent to a visual inspection. To understand the nature of timestamps in the field we explore six established recommendation datasets:

- **MovieLens 1M and 25M** : Our primary focus and one of the most widely used datasets for benchmarking recommendation performance. We explore both the ML-1M and ML-25M versions [34]. In these datasets the interactions represent a user rating a movie and the timestamps indicate when the rating was submitted to the nearest second.
- **Amazon Beauty**: A dataset with reviews of beauty products from *Amazon.com* introduced in [35]. The interactions are reviews of the products and the timestamps represent the date of review.
- **Amazon Video Games 2014 and 2018** : Datasets of video game products reviews. As both were available we look at the 2014 version introduced in McAuley et al. [35] and 2018 version introduced in Ni et al. [36]. As with Amazon Beauty, the interactions in this dataset represent a user reviewing an item and the timestamp is the date of the review.
- **Steam**: Introduced in Kang and McAuley [1], this dataset captures interactions between users and video games on *Steam*. The interactions are reviews of the video games, and the timestamps are the date of the review.

Table 1
Dataset statistics after pre-processing steps.

Dataset	Users	Items	Avg. Interactions Per User	Avg. Interactions Per Item	Interactions
Amazon Beauty 2014	22,363	12,101	8.9	16.4	0.2M
Amazon Video Games 2014	24,303	10,672	9.5	21.7	0.23M
Amazon Video Games 2018	50,688	16,898	9.0	26.9	0.45M
Steam	281,455	11,961	12.6	297	3.5M
MovieLens-1M	6,040	3,416	165.5	292.6	1.0M
MovieLens-25M	162,541	32,720	153.5	762.4	24.9M

For all datasets we apply the same pre-processing steps as detailed in Kang and McAuley [1] and Sun et al. [2] wherein we remove duplicate interactions and keep only users and items with at least 5 interactions. All datasets include timestamp data, and all datasets have a maximum resolution of days except for the two MovieLens datasets whose maximum resolution is at the level of seconds.

As seen in Table 1, ML-1M is the most dense of the datasets explored with the three Amazon datasets being the most sparse. ML-1M also has much longer interaction histories on average. This makes it of particular interest to researchers looking to explore longer range dynamics in user sequences for recommendation. However, as we show in the following sections, the use of MovieLens is especially problematic for evaluating sequentially driven methods.

3.1. Timestamps

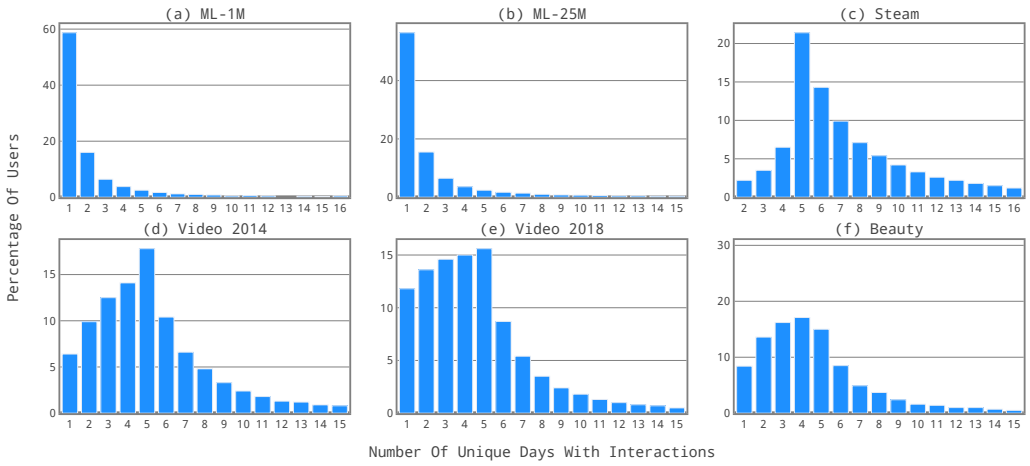

Figure 1: The percentage of users in each dataset whose interactions happened on x unique days.

In all datasets, each user history exists as a timestamp-ordered sequence of interactions with items. In this and the following sections we show that although the datasets include timestamps,

this does necessarily mean that these timestamps convey interaction order.

One way to clearly see the lack of true ordering in the ML-1M dataset is to look at the number of interactions for each user that happen at unique timestamps. Figure 1 shows the difference between the MovieLens datasets and others when viewed via the lens of interactions on unique days. For both MovieLens datasets the vast majority of users (59% and 56.4% for ML-1M and 25M respectively) have all their interactions occuring on a single date. When taken in conjunction with an average sequence length of over 150, these facts make it clear that the MovieLens datasets are not representative of a realistic sequence of movie watching. Steam is the only dataset with no users whose interactions are all on one day, and although there are single-day interaction sequences in the other datatsets, these users represent a much smaller percentage of the total users than they do in the MovieLens datasets.

The timestamp information in the MovieLens datasets is at the level of seconds, not days. Naturally then, it may be the case that although all interactions for a user occur on one or two days their ordering still provides some estimate of a genuine interaction history. Ideally, all users would have fully distinct interaction histories and therefore discernible order would exist between the interactions. However, this is not the case in MovieLens datasets with 0 second intervals accounting for 53.2% of timestamp intervals in ML-1M (17.4% in ML-25M). Further details are provided in Appendices A.1 & A.2. This further highlights the importance of dataset inspection and special care with regards to the application of timestamps for ordering in recommendation.

3.2. Intervals

In order to better understand the sequences in these datasets we look at each user's sequence of n interactions as a list of $n-1$ consecutive intervals. For example if a user interacted with item A on 2018-01-02 and item B on 2018-01-03 the interval would be one day. Some key statistics of this interval information are presented in Table 2. The mean-mode interval is the dataset-mean user-mode interval in days or seconds. The mode-mode interval statistic is the dataset-mode user-mode interval in days or seconds. We define the unique interaction ratio for a dataset as the number of interactions at distinct timestamps divided by the total number of interactions for each user averaged over the dataset. Of key importance is that if two interactions have an interval of zero, then the order in which those interactions occurred is unknowable.

Table 2
Interval statistics

Dataset	Mean-Mode Interval (Days/Seconds)	Mode-Mode Interval (Days/Seconds)	Unique Interaction Ratio (Days/Seconds)
Beauty 2014	4	0	0.61
Video 2014	9	0	0.69
Video 2018	7.8	0	0.6
Steam	14.7	0	0.86
ML-1M	~0 / 0.05	0 / 0	0.03 / 0.48
ML-25M	~0 / 3.24	0 / 0	0.04 / 0.8

As can be seen in Table 2, the mean-mode day-interval for ML-1M is nearly 0 over both days and seconds. This means that on average, users' interaction sequences contain more interactions happening at the same second as another interaction in the same user's sequence than not. Therefore the order of items in such a sequence is ill-posed and, given the time-scale under consideration, divorced from a realistic viewing pattern. Also of note is that for all datasets the mode-mode interval is 0 days or seconds. The significant amount of indistinct regions of interactions implied by this value means that interaction orders derived from all these datasets are contaminated with some amount of noise.

Figure 2 visualizes the percentage of intervals for each dataset that are of zero days in length. While all datasets have large amounts of zero-day intervals, ranging from 19.4% for Steam to 98.5% for ML-1M, the MovieLens datasets stand out as severely divergent from a real-world scenario. These distributions of timestamps do not reflect a natural interaction behavior where, for example, a person is unlikely to watch more than two or three movies in a single day. In fact the 59% of users in the ML-1M dataset whose entire interaction sequence is on a single day have a median sequence length of 62 interactions.

A large percentage of interactions in MovieLens share timestamps with 'adjacent' interactions. This behavior is problematic when using the ordering of these interactions as input information for modelling. Given these overlaps, the ordering only partly exists, and in the case of ML-1M mostly does not. See Appendix A.1 for further analysis.

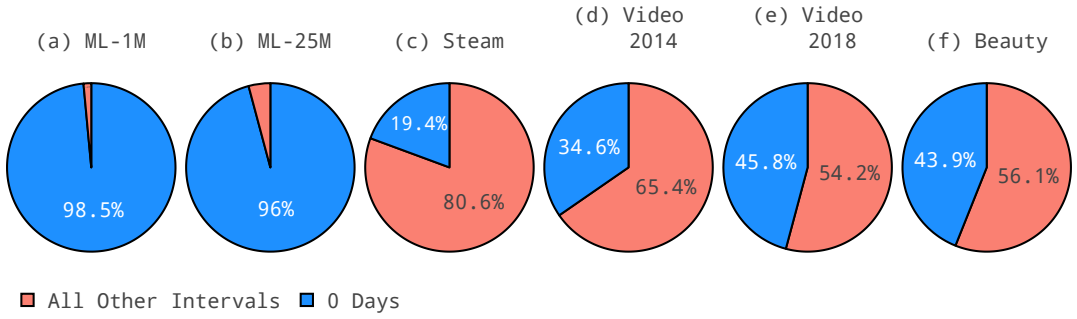

Figure 2: Percentage of intervals between consecutive interactions of length 0 days.

The data presented in Table 2 and Figure 2 raise interesting questions for the validity of these datasets as offline proxies for genuine sequential interaction data. By presenting Figures 1 and 2 as well as Table 2 we hope to convey that although the timestamp information in the MovieLens datasets is particularly problematic, the other datasets all suffer from the same affliction to lesser and varying degrees. We propose that the above analysis or something analogous to it becomes standard procedure for datasets when they are being used for evaluation sequential recommenders.

3.3. MovieLens for Sequential Recommendation

The MovieLens datasets are well established for evaluating recommendation engines [37, 38] and continue to be used by many of the leading models as one of the benchmarks for sequential recommendation [39, 1, 2, 40, 41, 42]. While the MovieLens datasets are incredibly valuable for assessing general recommendation algorithms, we find that they are not good datasets for assessing the performance of sequential recommendation. In fact, the originators of the datasets raise evidentiary concerns regarding the value of timestamps in their 2015 paper [34].

An argument could be made that although the ordered interaction data present in the dataset does not closely represent a real-world scenario, the sequences nonetheless do represent some estimate of order of interest. This is, however, in conflict with our findings that over 50% of interactions in user sequences from ML-1M have an equal timestamp as another interaction in the same user sequence. For ML-25M this percentage is 17.4%, showing that despite improvements to MovieLens, regions of ambiguous order still exist in a meaningful portion of the dataset.

It may be true that some user interactions that share timestamps were added to the dataset in such a way that the order of interaction was preserved. Following this, it may seem reasonable to infer that they present a valid source of order. However, if we sort the datasets by timestamp, as is often done during pre-processing, we are left to the whims of the sorting algorithm and how it decides to order the equal valued entries. This allows for the situation of two researchers using the same dataset to have different interaction orders for the same users. This problem could be remedied by an unambiguous 'interaction order' field included along side the timestamps at construction.

An additional factor mentioned by Harper and Konstan [34] is that the movies rated by users in the MovieLens datasets are often prompted by an internal recommendation engine. This means that the items users are served to rate depend on the items the user has previously rated. For the larger MovieLens datasets (10M, 20M and 25M) the interface and underlying recommendation engine have changed over the course of the dataset collection [34]. The presence of an internal recommendation engine introduces an implicit signal into a user's interaction sequence since it controls the scope of which items a user is served for rating at any given position in the sequence.

The examination of the datasets throughout this section points towards several issues with their applicability as evaluation benchmarks for sequential recommendation. In the following section we perform a set of experiments aimed to explore the breadth of impact of these pseudo-sequences on a sequential recommender model.

4. Experiments

We aim to explore the impact that the pervasive ordering pathologies of several common recommendation datasets have, if any, on sequential recommendation performance. Specifically, the goals of our experiments are to answer the following research questions: Given the questionable timestamp informed sequences in the datasets, how much impact does shuffling this information have on performance? Does the explicit construction by internal recommendation for MovieLens introduce a signal that sequential recommenders inadvertently exploit? If we adapt the experimental design to control for the conflated signal in MovieLens, what impact

does shuffling the data have on performance?

4.1. Implementation Details

We choose to use **SASRec** as the basis for our experiments because it is an established model with a left to right architecture whose training paradigm predicts the next item for each position in the sequence. We re-implement SASRec from the ground up in Tensorflow 2. For all datasets used we pull fresh copies from the sources[4]. We omit ML-25M from our experiments due to computational and time constraints, and leave this for future work. We adopt the same hyperparameters for each dataset as in Kang and McAuley [1].

Following the lead of Kang and McAuley [1], the metrics evaluated are Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG) [43] at 10. Evaluation is done by taking 100 items from the dataset that the user did not interact with and calculating relevance scores for these items. The relevance score for the held out item, either belonging to the validation or test set, is then compared to and ranked with the negative items. These metrics assume that the user has equal opportunity to pick any item in the dataset to interact with next. The MovieLens rating interface and internal recommendation engine explicitly invalidates this assumption.

4.2. Shuffled vs. Unshuffled

Our first experiment aims to determine the impact of explicitly randomizing the order of the training items for each user. We follow the training paradigm in Kang and McAuley [1]. The input for each user is the last N items in their history, where the second to last item is held out for validation and the last item is held out for test evaluation. To avoid biases from random seed selection, we perform twenty total runs for each dataset, ten shuffled and ten unshuffled with both sets sharing the same ten random seeds. In the shuffled cases we randomly re-order the users interaction history but keep the last two items untouched (for validation and test).

In Table 3 we report the results of our experiments as well as the reported results from Kang and McAuley [1]. Although all differences between shuffled and unshuffled results are statistically significant (except in the case of Steam where shuffling had no effect on performance) the differences are not qualitatively substantial outside of ML-1M where the ranges of shuffled and unshuffled are non-overlapping (and Beauty, though there the difference favors shuffled and will be explored briefly later). From a replication stand point our unshuffled results align well with the SASRec reported scores[5]. The focus of our subsequent analysis is on the difference in performance between the shuffled and unshuffled cases. The largest such difference on both HR@10 and NDCG@10 occurs for ML-1M. We propose that this difference is caused by the shuffling process destroying most of the detectable signal provided by the internal recommendation engine introduced by the MovieLens dataset construction.

Notably, shuffling has little effect on the performance of the other datasets. One factor that may explain how robust SASRec is to shuffling of the input sequences is that recommendations are made by looking at few items in the history due to the attentional mechanism [1]. Whether this robustness to input-sequence shuffling is a general property of sequential recommenders or

[4]We will provide all model and data cleaning code upon publication.

[5]There is a noticeable variation on the Steam dataset. We suspect this is due to differences in pre-processing.

Table 3
Mean scores over ten random initializations. All runs use the same parameters as in [1] and were ran for 200 epochs.

Dataset	Metric	SASRec	Unshuffled	Shuffled	Delta	Percentage
ML-1M	HR@10	0.825	0.814	0.681	-0.133	-16.3%
	NDCG@10	0.591	0.583	0.408	-0.175	-30.0%
Video Games 2014	HR@10	0.741	0.729	0.715	-0.014	-2.0%
	NDCG@10	0.536	0.506	0.486	-0.020	-4.0%
Video Games 2018	HR@10	-	0.710	0.705	-0.005	-0.7%
	NDCG@10	-	0.494	0.482	-0.012	-2.5%
Steam	HR@10	0.873	0.836	0.839	0.003	0.4%
	NDCG@10	0.631	0.587	0.586	-0.001	-0.2%
Beauty	HR@10	0.485	0.484	0.508	0.024	5.0%
	NDCG@10	0.322	0.325	0.332	0.007	2.0%

is specific to attention based sequential recommenders exclusively is an interesting question and one we leave for future work. Another factor may be in the dataset construction itself, given that the other datasets come from processes that better proxy a realistic interaction scenario than MovieLens.

Although small, the difference in NDCG@10 for Video 2014, Video 2018 and Beauty are all statistically significant. While shuffling negatively impacts the performance for the two Video datasets, interestingly, shuffling seems to improve the performance on Beauty. Statistically, Beauty is not too dissimilar to Video 2018 as shown in Section 3. We propose that the difference in response to shuffling may be due to domain specific differences in how users interact with items. Analysis of the loss characteristics for these runs suggest that shuffling the Beauty dataset improves generalization of the model. Learning is slower but eventually crosses the plateau reached by the model on the unshuffled data.

4.3. Rating Prediction Experiments

For this experiment we modify SASRec to predict the ratings of the held-out items rather than predicting the items themselves. Our goal is to remove the impact of the internal recommendation engine in ML-1M by limiting the scope of the model to only items that users interacted with. This separation of ratings from item identification allows for a more fair comparison between the shuffled and unshuffed cases. That is, by asking the model to rate items rather than suggest items, we remove the benefit of being able to predict which items the internal recommendation engine would have suggested.[6] We modify the loss function of SASRec to have two parts, one mean squared error component that penalizes distance from the true rating for predictions and one cross-entropy component that penalizes wrong labels.

Table 4 show the results of shuffled and unshuffled runs on ML-1M for the rating prediction

[6]It has been noted that rating prediction is a poor measure for recommendation algorithm evaluation [44]. We nonetheless find it useful for teasing apart our confounded data-sequence.

Table 4
Results of the rating prediction experiment. Delta is this difference between shuffled and unshuffled. The relative performance drop reframes delta as a percentage of the unshuffled score for a given metric.

Holdout N	Metric	Unshuffled	Shuffled	Delta	Relative Performance Drop
3	Accuracy	0.416	0.411	0.005	-1.2%
	RMSE	1.156	1.168	-0.012	-1%
1	Accuracy	0.421	0.412	0.009	-2.1%
	RMSE	1.141	1.167	-0.026	-2.3%

version of SASRec. We present accuracy and root mean squared error (RMSE) for two values of holdout N, this value represents the number of items held out for test and validation. By changing the evaluation method to one that does not depend on scores for items with which the user did not interact we are able to close the performance gap considerably, from tens of percent to single digits. These results provide further evidence that the difference in performance on the ML-1M dataset between shuffled and unshuffled shown in Section 4.2 is due to a sequential bias that constrains the possibility space of items that can appear later in a user's sequence. That being the case, this suggests that a model's ability to utilize the order information in ML-1M does not translate beyond the dataset and thus ML-1M is an especially poor benchmark for sequential recommenders in general.

5. Discussion

We have shown that the MovieLens datasets, while important contributions and useful benchmarks for recommendation in general, are inappropriate for use in the subfield of sequential recommendation. Of note is that, while the presence of a sequential signal to the data implies that MovieLens is an effective benchmark for sequential pattern recognition, the detachment of that signal from real-world watch habits makes it unsuitable as a benchmark for sequential recommendation specifically. That is, any pattern embedded into a sequence of data would make a valid test case for sequential pattern recognition, but recommendation, while related, has different constraints and more specific goals. Facts pertaining to these issues were acknowledged in the original release of the datasets, but seem to be often overlooked and/or forgotten in the field. We performed a rigorous analysis on two of the MovieLens datasets and compared them to other benchmark datasets to further elucidate the inherent issues therein. Though the MovieLens datasets clearly stood-out, the other datasets all contain significant amounts of ratings with indistinct timestamps for users as well, suggesting that ordering issues may be generally pervasive across benchmarks in the field.

To directly examine the impact of ordering information in these datasets for sequential recommendation we conducted the following two experiments. The first explicitly destroyed any sequential information in the datasets by randomly shuffling the training sequence. The difference between shuffled and unshuffled was found to be small in most cases with the notable exception being ML-1M. Shuffling caused a large drop in performance for ML-1M, which we

propose is a symptom of the dataset's construction, namely the contribution to a sequential signal introduced by the internal recommendation engine. Our second experiment aimed to remove the impact from this internal recommendation system by attempting to predict the rating given by users for items they interacted with. We showed that doing this greatly reduced the performance gap between shuffled and unshuffled, providing further evidence that a large chunk of the performance of SASRec on ML-1M comes from modelling the internal recommendation engine of the MovieLens system and not genuine sequential information.

Assessing the value of offline testing requires a precise understanding of the differences between benchmarks and the real-world scenarios we aim to emulate. What constitutes a true sequence looks different depending on the domain. For example, you may purchase multiple beauty items at the same time but you are unlikely to watch more than one movie at once. Furthermore, when the timestamp and ordering information is distanced from a natural interaction sequence, the ability of sequential recommendation models to generalize is called into question. Our evaluation does not speak to the performance of such algorithms, rather that the true performance may be obscured by a lack of appropriate datasets.

6. Conclusion

Our work highlights the importance of further scrutiny for dataset creation methodologies when using those datasets as benchmarks for tasks beyond their initial scope. Specifically, we find there is a necessity for further exploration and creation of new datasets for evaluating sequential recommendation, with special attention paid to temporal tagging and order information.

Acknowledgments

References

[1] W. Kang, J. McAuley, Self-attentive sequential recommendation, in: 2018 IEEE International Conference on Data Mining (ICDM), IEEE Computer Society, Los Alamitos, CA, USA, 2018, pp. 197–206. URL: https://doi.ieeecomputersociety.org/10.1109/ICDM.2018.00035. doi:10.1109/ICDM.2018.00035.

[2] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, P. Jiang, Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1441–1450. URL: https://doi.org/10.1145/3357384.3357895. doi:10.1145/3357384.3357895.

[3] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in: Proceedings of the 19th International Conference on World Wide Web, WWW '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 811–820. URL: https://doi.org/10.1145/1772690.1772773. doi:10.1145/1772690.1772773.

[4] R. He, J. McAuley, Fusing similarity models with markov chains for sparse sequential

recommendation, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, IEEE, Barcelona, Spain, 2016, pp. 191–200.

[5] A. Torralba, A. A. Efros, Unbiased look at dataset bias, in: CVPR 2011, IEEE, Colorado Springs, CO, USA, 2011, pp. 1521–1528. doi:10.1109/CVPR.2011.5995347.

[6] T. Tommasi, N. Patricia, B. Caputo, T. Tuytelaars, A Deeper Look at Dataset Bias, Springer International Publishing, Cham, 2017, pp. 37–55. URL: https://doi.org/10.1007/978-3-319-58347-1_2. doi:10.1007/978-3-319-58347-1_2.

[7] H. He, S. Zha, H. Wang, Unlearn dataset bias in natural language inference by fitting the residual, in: Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 132–142. URL: https://www.aclweb.org/anthology/D19-6115. doi:10.18653/v1/D19-6115.

[8] T. Sun, A. Gaut, S. Tang, Y. Huang, M. ElSherief, J. Zhao, D. Mirza, E. Belding, K.-W. Chang, W. Y. Wang, Mitigating gender bias in natural language processing: Literature review, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 1630–1640. URL: https://www.aclweb.org/anthology/P19-1159. doi:10.18653/v1/P19-1159.

[9] C. G. Northcutt, A. Athalye, J. Mueller, Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks, 2021. arXiv:2103.14749.

[10] R. Li, J. S. Johansen, H. Ahmed, T. V. Ilyevsky, R. B. Wilbur, H. M. Bharadwaj, J. M. Siskind, The perils and pitfalls of block design for eeg classification experiments, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (2021) 316–333. doi:10.1109/TPAMI.2020.2973153.

[11] P. Kouki, I. Fountalis, N. Vasiloglou, X. Cui, E. Liberty, K. Al Jadda, From the lab to production: A case study of session-based recommendations in the home-improvement domain, in: Fourteenth ACM Conference on Recommender Systems, RecSys '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 140–149. URL: https://doi.org/10.1145/3383313.3412235. doi:10.1145/3383313.3412235.

[12] A. Gruson, P. Chandar, C. Charbuillet, J. McInerney, S. Hansen, D. Tardieu, B. Carterette, Offline evaluation to make decisions about playlistrecommendation algorithms, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 420–428. URL: https://doi.org/10.1145/3289600.3291027. doi:10.1145/3289600.3291027.

[13] M. Quadrana, P. Cremonesi, D. Jannach, Sequence-aware recommender systems, ACM Computing Surveys (CSUR) 51 (2018) 1–36.

[14] P. G. Campos, F. Díez, I. Cantador, Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols, User Modeling and User-Adapted Interaction 24 (2014) 67–119.

[15] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, WWW '01, Association for Computing Machinery, New York, NY, USA, 2001, p. 285–295. URL: https://doi.org/10.1145/371920.372071. doi:10.1145/371920.372071.

[16] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, Using sequential and non-sequential patterns in predictive web usage mining tasks, in: 2002 IEEE International Conference on Data

Mining, 2002. Proceedings., IEEE, IEEE, Maebashi City, Japan, 2002, pp. 669–672.
[17] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based recommendations with recurrent neural networks, 2015.
[18] M. Quadrana, A. Karatzoglou, B. Hidasi, P. Cremonesi, Personalizing session-based recommendations with hierarchical recurrent neural networks, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 130–137. URL: https://doi.org/10.1145/3109859.3109896. doi:10.1145/3109859.3109896.
[19] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, D. Cai, What to do next: Modeling user behaviors by time-lstm, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, AAAI Press, Palo Alto, California, 2017, p. 3602–3608.
[20] P. Covington, J. Adams, E. Sargin, Deep neural networks for youtube recommendations, in: Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 191–198. URL: https://doi.org/10.1145/2959100.2959190. doi:10.1145/2959100.2959190.
[21] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, J. Wu, Sequential recommender system based on hierarchical attention network, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, AAAI Press, Palo Alto, California, 2018, p. 3926–3932.
[22] N. Ferro, D. Kelly, Sigir initiative to implement acm artifact review and badging, SIGIR Forum 52 (2018) 4–10. URL: https://doi.org/10.1145/3274784.3274786. doi:10.1145/3274784.3274786.
[23] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, K. Crawford, Datasheets for datasets, 2018.
[24] S. Holland, A. Hosny, S. Newman, J. Joseph, K. Chmielinski, The dataset nutrition label: A framework to drive higher data quality standards, 2018.
[25] E. M. Bender, B. Friedman, Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science, Transactions of the Association for Computational Linguistics 6 (2018) 587–604. URL: https://doi.org/10.1162/tacl_a_00041. doi:10.1162/tacl_a_00041.
[26] S. Rendle, L. Zhang, Y. Koren, On the difficulty of evaluating baselines: A study on recommender systems, 2019.
[27] M. D. Ekstrand, Lenskit for python: Next-generation software for recommender systems experiments, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 2999–3006. URL: https://doi.org/10.1145/3340531.3412778. doi:10.1145/3340531.3412778.
[28] Z. Gantner, S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Mymedialite: A free recommender system library, in: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 305–308. URL: https://doi.org/10.1145/2043932.2043989. doi:10.1145/2043932.2043989.
[29] Z. Sun, D. Yu, H. Fang, J. Yang, X. Qu, J. Zhang, C. Geng, Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison, in:

Fourteenth ACM Conference on Recommender Systems, RecSys '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 23–32. URL: https://doi.org/10.1145/3383313.3412489. doi:10.1145/3383313.3412489.

[30] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, K. Li, Y. Chen, Y. Lu, H. Wang, C. Tian, X. Pan, Y. Min, Z. Feng, X. Fan, X. Chen, P. Wang, W. Ji, Y. Li, X. Wang, J.-R. Wen, Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms, 2020.

[31] V. W. Anelli, A. Bellogín, A. Ferrara, D. Malitesta, F. A. Merra, C. Pomo, F. M. Donini, T. Di Noia, Elliot: a comprehensive and rigorous framework for reproducible recommender systems evaluation, 2021. arXiv:2103.02590.

[32] A. Said, A. Bellogín, Comparative recommender system evaluation: Benchmarking recommendation frameworks, in: Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 129–136. URL: https://doi.org/10.1145/2645710.2645746. doi:10.1145/2645710.2645746.

[33] Y. Ji, A. Sun, J. Zhang, C. Li, A critical study on data leakage in recommender system offline evaluation, 2021. arXiv:2010.11060.

[34] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, Acm transactions on interactive intelligent systems (tiis) 5 (2015) 1–19.

[35] J. McAuley, C. Targett, Q. Shi, A. van den Hengel, Image-based recommendations on styles and substitutes, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 43–52. URL: https://doi.org/10.1145/2766462.2767755. doi:10.1145/2766462.2767755.

[36] J. Ni, J. Li, J. McAuley, Justifying recommendations using distantly-labeled reviews and fine-grained aspects, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 188–197. URL: https://www.aclweb.org/anthology/D19-1018. doi:10.18653/v1/D19-1018.

[37] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, p. 689–698. URL: https://doi.org/10.1145/3178876.3186150. doi:10.1145/3178876.3186150.

[38] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2017, p. 173–182. URL: https://doi.org/10.1145/3038912.3052569. doi:10.1145/3038912.3052569.

[39] Q. Liu, S. Wu, D. Wang, Z. Li, L. Wang, Context-aware sequential recommendation, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, Barcelona, Spain, 2016, pp. 1053–1058. doi:10.1109/ICDM.2016.0135.

[40] A. Yan, S. Cheng, W.-C. Kang, M. Wan, J. McAuley, Cosrec: 2d convolutional neural networks for sequential recommendation, in: Proceedings of the 28th ACM International

Conference on Information and Knowledge Management, CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 2173–2176. URL: https://doi.org/10.1145/3357384.3358113. doi:10.1145/3357384.3358113.

[41] P. Zhao, T. Shui, Y. Zhang, K. Xiao, K. Bian, Adversarial oracular seq2seq learning for sequential recommendation, in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, International Joint Conferences on Artificial Intelligence Organization, San Francisco, CA, USA, 2020, pp. 1905–1911. URL: https://doi.org/10.24963/ijcai.2020/264. doi:10.24963/ijcai.2020/264, main track.

[42] M. Ji, W. Joo, K. Song, Y.-Y. Kim, I.-C. Moon, Sequential recommendation with relation-aware kernelized self-attention, Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 4304–4311. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5854. doi:10.1609/aaai.v34i04.5854.

[43] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Transactions on Information Systems (TOIS) 20 (2002) 422–446.

[44] S. M. McNee, J. Riedl, J. A. Konstan, Being accurate is not enough: How accuracy metrics have hurt recommender systems, in: CHI '06 Extended Abstracts on Human Factors in Computing Systems, CHI EA '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 1097–1101. URL: https://doi.org/10.1145/1125451.1125659. doi:10.1145/1125451.1125659.

A. Appendices - Additional Figures

A.1. Indistinct Sequences

Figure A.1 (a) shows that the user sequences in ML-1M are particularly contaminated with large regions of indistinct interactions. These regions represent areas where the actual interaction order is unknowable, and thus unusable by sequential recommenders. While less pronounced, this issue persists for ML-25M.

A.2. Per Second Data

The timestamps associated with interactions in the MovieLens datasets are distinct to the level of seconds. Figure A.2 shows both the percentages of zero second intervals in the ML-1M (a) and ML-25M (b) datasets as well as the cumulative percentage of users plotted against the number of interactions with unique timestamps. Figure A.2 (c) shows that over two-thirds of the users in both datatsets have interaction histories containing less than 100 unique timestamps. Only 2 of the 6040 users in ML-1M have all their interactions at distinct timestamps. The average interaction history contains 165.5 and 153.5 interactions for ML-1M and ML-25M respectively.

A.3. Shuffled Experiment Visualization

Figure A.3 shows the performance difference between shuffled and unshuffled runs. Figure A.3 (a) shows a significantly larger difference than the others, illustrating presence of a strong sequential signal in ML-1M.

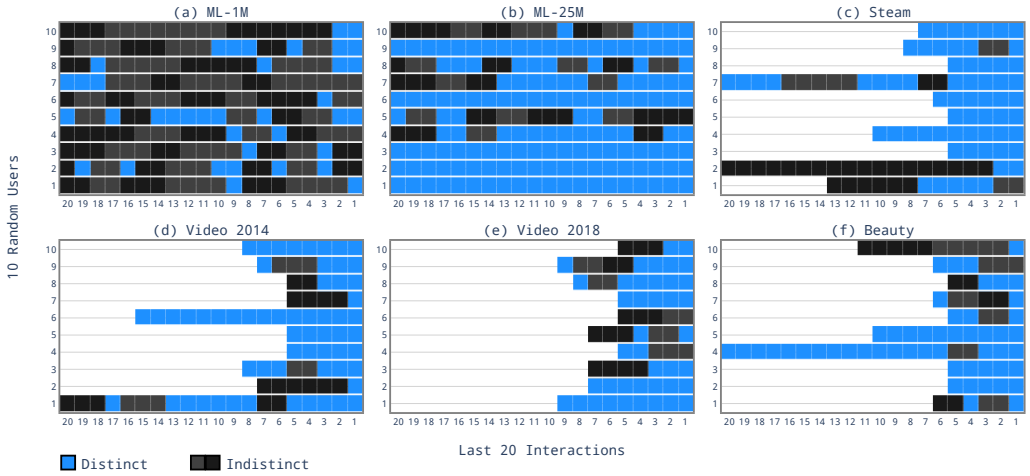

Figure A.1: The last 20 interactions for 10 randomly selected users for each of the six datasets. Each row represents a single user's last 20 interactions and each box represents a single interaction. All interactions with a unique timestamp are presented in blue. Interactions that share a timestamp with a neighboring interaction are colored one of two shades of grey. We use the two shades to separate neighboring chunks of same-timestamp interactions. Uncolored sections represent users with fewer than 20 total interactions.

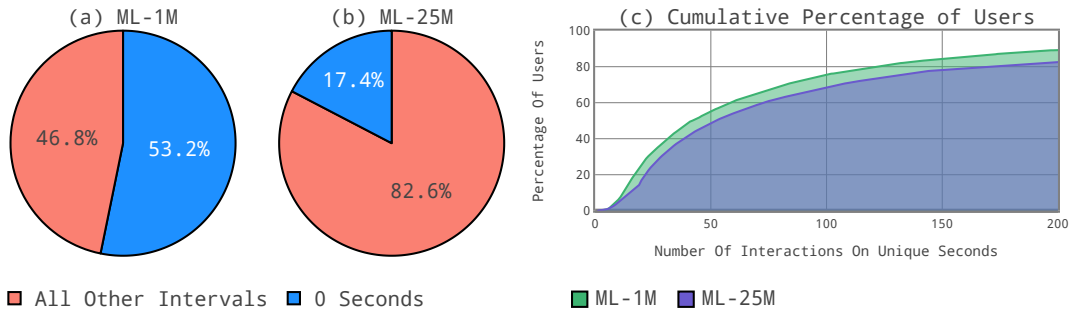

Figure A.2: a, b: Percentage of intervals between consecutive interactions of length 0 seconds for ML-1M and ML-25M respectively. c: Cumulative percentage of users in each dataset whose interactions happened on X unique timestamps.

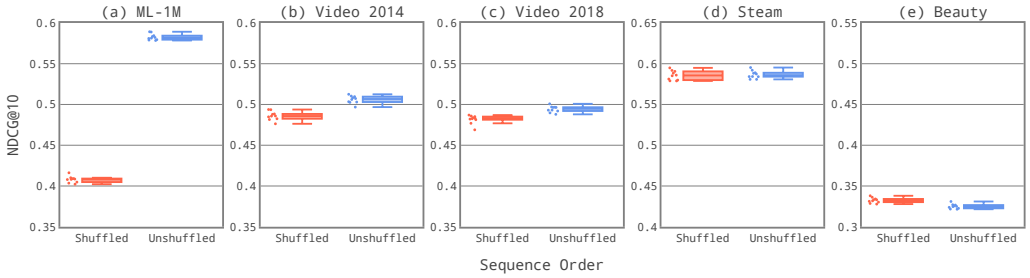

Figure A.3: NDCG@10 results of ten separate runs on each of the datasets. Points show the distribution of scores, with box-plots to show the inter-quartile range and whiskers spanning the minimum and maximum values. Red points and box-plots are shuffled results and blue represent unshuffled. Though overlapping, only the difference between shuffled and unshuffled for Steam is not statistically significant.