Fair Offline Evaluation Methodologies for Implicit-Feedback Recommender Systems with MNAR Data

Olivier Jeunen
University of Antwerp
olivier.jeunen@uantwerp.be

Koen Verstrepen
Froomle
koen.verstrepen@froomle.com

Bart Goethals
University of Antwerp, Froomle,
Monash University
bart.goethals@uantwerp.be

ABSTRACT

Recommender systems are traditionally evaluated using historical data, partitioned into a training- and test-set. The system is trained on the user-item interactions available in the training set and evaluated on its performance to predict which interactions are part of the test set. Leave-One-Out Cross-Validation (LOOCV) is a commonly recurring evaluation procedure, widely used to present novel algorithms as the state-of-the-art. However, the temporal aspect that is inherent to many recommender system use cases is entirely neglected with this technique, as well as potential biases in the data (i.e. interactions are Missing-Not-At-Random (MNAR)).

In this paper we propose and experimentally validate an alternative method to perform offline evaluation using real-world data from a live recommender system. Our novel approach adheres to the aspects that are inherent to web-based recommender systems in e.g. e-commerce much more tightly than LOOCV. Experimental results indicate that LOOCV is prone to overestimate model performance in general, underestimate the power of popularity-based baselines, and generally rank algorithms differently than our methodology. Furthermore, we study the impact of live recommendation algorithms in place during the time of data gathering on the offline evaluation of other algorithms on said data. We experimentally validate that such impact is indeed significant. Finally, we propose a scope for future research to model these MNAR biases and take them into account during training and evaluation to provide unbiased recommendations.

CCS CONCEPTS

•Information systems → Recommender systems; Evaluation of retrieval results; *Collaborative filtering;* **•Computing methodologies** → Learning from implicit feedback;

KEYWORDS

Recommender systems, offline evaluation, missing-not-at-random

ACM Reference format:
Olivier Jeunen, Koen Verstrepen, and Bart Goethals. 2018. Fair Offline Evaluation Methodologies for Implicit-Feedback Recommender Systems with MNAR Data. In *Proceedings of the REVEAL '18 Workshop on Offline Evaluation, Vancouver, Canada, October 2018 (REVEAL '18),* 8 pages.
DOI: N.A.

REVEAL '18, Vancouver, Canada

DOI: N.A.

1 INTRODUCTION

Over the past decades, personalisation has played an ever-growing role in how we consume content online. News websites, movie or music streaming services, retail stores, etc. can all greatly benefit from recommendation systems that accurately pair items with users and suggest them. Users can be guided towards the subset of catalogued items they are interested in, leading to more satisfying experiences and an increase in user engagement.

When recommender systems first gained traction, the field focused on the task of rating prediction. This assumes that a dataset with explicit feedback from users is available, which is often hard to collect. The goal was then to predict users' ratings for unseen items, with the rationale that items with higher predicted ratings make up better recommendations. In recent years, a shift has occurred towards item prediction from implicit feedback. These methods do not require explicit ratings by users, but rather take into account logged interactions between users and items to model inherent preferences and correlations. Throughout our work, we will focus on this task, as implicit feedback data is more prevalent in present day e-commerce.

Specific goals of recommender systems can vary greatly depending on their respective applications. Where some systems will be more focused on maximising user engagement in terms of time spent browsing a website, others might only focus on clicks or sales. User satisfaction, serendipity or diversity of the recommended items are only a few examples of many more possible objectives. We focus on maximising user engagement through clicks for the rest of this work, but our methodology is easily extended towards maximising sales. Analogously, this work focuses on, but is not limited to, collaborative filtering (CF) algorithms.

Traditionally, the performance of recommender systems (learning from implicit feedback) is evaluated on historical transactional data. As is often the case with classification problems and supervised learning in general, a portion of the data is split off and used as a test or validation set to assess algorithmic performance [15]. Leave-One-Out Cross-Validation (LOOCV) is a commonly recurring technique in the literature, where for every user one item-interaction is randomly selected to be part of the test set. The training set then consists of all remaining user-item pairs. Common metrics such as the *hit-rate-at-k* (HR@k) then compute the fraction of users for whom the removed item occurs in the top-k recommendations computed by the system, or the *normalised discounted cumulative gain* (NDCG@k) which evaluates the ranking of the removed item in the recommendation-list, or others. This process is repeated with different training-test splits, and performance metrics are subsequently aggregated over different runs in order to get a stable final result. Algorithms that can generate more

promising metrics are then assumed to generate more revenue in an on-line setting than their competitors. This technique has been used widely and recently to present new algorithms as the state-of-the-art [4, 8, 16–18, 21].

On-line evaluation methods such as live A/B-testing have been shown to paint a clearer and more honest image of an algorithm's performance in providing meaningful and interesting recommendations, but are generally more expensive and complex to realise [25]. During an A/B-test, the user-base is divided into groups. Every user in one of those groups is presented with recommendations generated by a different algorithm, specific to the group they belong to. Metrics such as the *click-through-rate* (CTR) are then often used to compare which algorithm generates the most clicks, and thus is the most successful in generating revenue. Online evaluation is often favoured over its offline counterpart, as it directly correlates with the inherent goal of the system: to maximise user-interaction with the shown recommendations.

Crucial differences between live A/B-tests and LOOCV are two-fold: first there is a clear temporal dimension in many recommender systems use cases that is inherently taken into account in the first, but often neglected in the latter. Predicting past preferences based on future interactions is in many cases a considerably easier task than vice versa, and as a consequence their performances are not necessarily representative for each other.

Second, the goal of the evaluation technique is inherently dissimilar; where A/B-tests can present and evaluate a wide range of recommendations and evaluate how the user interacts with them, LOOCV is entirely restricted to predicting which items the user has already interacted with in the past. A top-k recommendation list might be clicked in an A/B-test because it sparks the user's interest, but it won't increase the hit-rate of LOOCV if the user has no recorded interactions with these items in the historical dataset. This distinction between negative and missing feedback is taken into account in the training phase of several well-known algorithms [11, 19, 21], but is much less studied in the context of offline evaluation.

Furthermore, if the available historical user-item interactions were collected with a live recommender system in place, this indicates the data are Missing-Not-At-Random (MNAR) [27]. We show that the algorithm presenting recommendations to the user significantly impacts offline evaluation results of different algorithms on the collected data. We adopt the terminology used by recent work in counterfactual estimation for recommender system evaluation and call such a live recommender system algorithm the *logging policy*. These counterfactual estimators have been shown to act as fast offline alternatives for more classical online methods such as A/B-testing, and are proven to be more correlated with online metrics than classical offline alternatives. As a consequence, improving these estimators has become a lively research direction in recent years [2, 6].

Related work has studied the correlation of the above-mentioned off- and on-line evaluation methods. Comparison research studies do exist for the specific fields of research paper recommendation [3], movie recommendation [22] and news recommendation [5], but none for the more general case. What these studies have in common however, is that they shed doubt on the assumption being made in many research papers that off-line evaluations are good indicators of on-line performance. A discussion of how recommendation systems should be evaluated in an offline manner is presented by Herlocker et al. [10]. The authors identify a range of different goals the recommendation system and subsequently the evaluation method might need to be tuned to. A comparison of various evaluation metrics suggests that different metrics can be highly uncorrelated and make the evaluation procedure even less deterministic.

The contributions presented in this paper are the following:

(1) We present a novel offline evaluation procedure that is more tightly coupled with the inherent goals of live recommender systems, and call it SW-EVAL.
(2) We show that SW-EVAL generates very different results than LOOCV in terms of absolute metrics, ratios among algorithms and rankings among algorithms. We experimentally validate our findings on real-world data from a live e-commerce recommender system.
(3) We show that the algorithm behind the live recommender system (or logging policy) induces a significant bias on collected data and as a consequence, severely influences offline evaluation results on said data.

The rest of this paper is structured as follows: we provide an overview of our alternative evaluation strategy in Section 2, and motivate our research questions. The data and algorithms we used for our experiments are presented and discussed in Section 3, along with their results. Our work is concluded in Section 4, where we finalise with a scope for future research.

2 METHODOLOGY

In what follows, we provide an overview of our methods. The following subsection focuses on preliminaries, after which we present our evaluation procedures and metrics. We then go on to motivate the research questions we aim to answer with this work, and how we achieve this.

2.1 Preliminaries

Throughout this paper, we assume to work with a set of historical transactional data, containing de-duplicated and timed logs of user-item interactions. U denotes the set of m unique users appearing in the dataset, and I the set of n unique items. A transaction is represented as a tuple $(u, i, t) \in U \times I \times \mathbb{R}^+$ where u is a user, i an item, and t a timestamp. $\mathcal{D}$ is the set of all available transactions. These transactions denote that user u has in some way consumed or interacted with item i at time t, be it in the form of a product purchase, a movie streaming, a click on a news article or otherwise. We represent these interactions in the form of a sparse user-item matrix $\mathbf{R} \in \{0, 1\}^{m \times n}$, often called the rating or preference matrix. Rows in this matrix are users represented by the items they have consumed, and vice versa for columns: $\mathbf{R}_{u,i} = 1$ if and only if user u has consumed item i and $\mathbf{R}_{u,i} = 0$ otherwise. For a given item i, we define the set U_i as consisting of all users u that have consumed item i, that is $U_i = \{u \in U : \mathbf{R}_{u,i} = 1\}$, and I_u analogously for a given user u: $I_u = \{i \in I : \mathbf{R}_{u,i} = 1\}$. When we represent an item i or a user u as their respective column- or row-vectors in $\mathbf{R}$, we write them as $\vec{i}$ or $\vec{u}$. Time-intervals are characterised by subscripts: $\mathcal{D}_t$ is the set of all interactions up to but not including time t, $\mathbf{R}_{[t_x, t_y)}$

is the preference matrix containing all transactions $(u, i, t) \in \mathcal{D}$ where $t_x \leq t < t_y$. The timestamp of the latest transaction in $\mathcal{D}$ is denoted by $t_{\max}$.

Finally, as logged interactions in our setting originate from a system running live A/B-tests, we distinguish $\mathcal{D}^\pi$ as the set of transactions generated under logging policy π. In the trivial case where only one logging policy π is implemented, $\mathcal{D} = \mathcal{D}^\pi$. Note that sets of transactions generated under different logging policies typically contain disjoint sets of users, but the same sets of items.

2.2 Evaluation Procedure

As we have mentioned in Section 1, LOOCV is one of the most commonly recurring evaluation techniques in recommender system literature. For every user u, one item j is sampled uniformly at random from I_u to be used in the validation phase; all other remaining item-interactions $\{(u, i, t) \in \mathcal{D} : i \in I_u \backslash \{j\}\}$ are used to train the model and generate predictions. The model is then evaluated on its ability to predict which item was left out. This process is repeated several times with different random seeds, and results are then aggregated over runs. In this way the approach generates statistically stable results, covering every user in the dataset. We argue that what impedes this approach from being a good proxy for online behaviour, is that it completely ignores the chronological ordering of events. Not only does the model use future interactions to predict past interactions for a given user, future information about item correlations will be used as well to predict an interaction at a given earlier time.

As all transactions are timed, the fairest method to perform the train-test split would be a hard cut on a certain timestamp t: all interactions in $\mathcal{D}_t$ are used for training, and all interactions in $\mathcal{D}_{[t, t_{\max}]}$ can be used for testing. This process can be repeated, splitting on different times t and aggregating results in order to get statistically stable results over the full dataset. It is important to note that not all users are included in the evaluation at every split: to properly evaluate a user u, she should have recorded historical interactions to base recommendations on (we do not consider the extreme cold-start case [24]), and future interactions to predict and evaluate on. As this set of users might possibly be very small at certain points, we do not incorporate those where $|U_t \cap U_{[t, t_{\max}]}| < u_{min}$, with u_{min} a predefined threshold. If the number of users used for evaluation is too small, outliers will have an overly large impact on the overall view.

Furthermore, live recommender systems are not as static as they are made out to be in LOOCV. These systems are either incrementally trained or fully retrained regularly, with possibly multiple updates every hour. As a consequence, specific to the use case, it might not be best practice to evaluate a system on its ability to predict relevant items multiple weeks or even months in the future. In our proposed evaluation procedure, we divide the dataset into equidistant intervals of width t_Δ. At a given time t, we evaluate the model trained on $\mathcal{D}_t$ on its ability to generate relevant recommendations w.r.t. the interactions present in $\mathcal{D}_{[t, t+t_\Delta)}$. The granularity of t_Δ is relative to the use case. Where it might be very small for a news recommender (new items arrive at high rates and item popularity generally declines quickly over time, calling for fast and frequent updates), retail recommenders might call for wider intervals (daily or weekly, as sales are bound to seasonality).

Throughout the rest of this paper, we refer to our novel proposed procedure as k-fold Sliding Window Evaluation (SW-EVAL) where k indicates the number of intervals used in the validation step.

Our method corresponds to live A/B-testing in the sense that it follows a clear chronological ordering of events, which we argue is crucial to properly assess system performance. A major difference that remains is that of an over-representation of false negatives during the evaluation phase: where in a real-time setting a user might click on a certain recommendation when it would be given, that same recommendation will always be seen as non-relevant by offline evaluation procedures if there exists no historical interaction between said user and item. As this issue is very non-trivial to solve, a careful choice of evaluation metrics that focus on rewarding true positives instead of penalising false positives is appropriate. We provide a brief overview of such metrics in the following subsection.

2.3 Evaluation Metric

As mentioned above, due to the inherent lack of user interaction in offline evaluation, we focus on metrics that reward true positives instead of those that penalise false positives. We will denote the set of known relevant items for a given user u as Rel_u, where her top-k recommendations are represented by $\text{Rec}_{u,k}$. Recall@k is then given by equation 1.

$$\text{Recall@}k = \frac{1}{m} \sum_{u \in U} \frac{|\text{Rel}_u \cap \text{Rec}_{u,k}|}{|\text{Rel}_u|} \tag{1}$$

With the leave-one-out scheme, Rel_u will always consist of exactly one item. In these cases, Recall@k is the same as HR@k: the fraction of users for whom the left-out item appears in the top-k recommendations.

In the more general case, Recall@k is the average fraction of retrieved relevant items in the top-k recommendations of all users. Note that when the number of relevant items is higher than k, it is impossible to achieve the perfect recall of 1. When k is set to the number of relevant items for every user $|\text{Rel}_u|$, equation 1 is called the R-precision, overcoming said issue.

2.4 LOOCV vs. SW-EVAL

The first hypothesis to tackle is whether LOOCV and SW-EVAL produce *comparable* results. We define *comparable* by three criteria in increasing order of importance:

Absolute metrics. One of the main goals of recommender systems evaluation is to obtain a reliable estimate of the effectiveness of a recommendation algorithm. The order of magnitude of recommendation accuracy is therefore of critical importance. Business trade-offs that require accurate estimations of e.g. model cost vs. model performance cannot afford large errors here.

Ratios among algorithms. As recommendation accuracy is often only one aspect of a broader evaluation, ratios among algorithms should be accurate as well. From an offline evaluation procedure, some model A might outperform some model B with a factor of 10. However, model A might also be 5 times more costly. If in practice,

model A would only be twice as effective as model B, the inaccurate evaluation procedure leads to suboptimal decisions.

Rankings among algorithms. In the case where model efficiency and cost are not taken into account, the ranking of competing algorithms that emerges from a certain evaluation procedure is still highly important. If simply the highest ranking model according to offline tests is deployed, it is imperative that the optimal model according to the offline evaluation procedure indeed reflects the best performer in an online setting as well. Naturally, an offline evaluation procedure that correlates well with the system's actual online application is preferred.

As we motivated our evaluation procedure to be tightly coupled with the inherent characteristics of live recommender systems in e-commerce and other use cases, in the case of conflicting results on any of the above-mentioned aspects, we would place more trust in SW-EVAL than in LOOCV. To validate the effectiveness of SW-EVAL compared to LOOCV, we aim to study the parity between results of these offline evaluation procedures with those attained through live A/B-tests in future work.

2.5 Impact of Logging Policy

A second hypothesis we aim to investigate with this work is whether the logging policy π has significant impact when evaluating results on $\mathcal{D}^\pi$ or more generally on any $\mathcal{D}$ s.t. $\mathcal{D}^\pi \subseteq \mathcal{D}$. The work of Agarwal et al. provides a theoretical foundation for this problem in the more general case, where data from multiple diverging stochastic logging policies is naively combined [2]. However, to the best of our knowledge, in the context of recommender systems, no studies have conclusively proven or disproven that the impact of π is indeed highly significant with real-world data. If this is indeed the case, interesting directions for future research include modelling the bias π incurs, and deriving learning algorithms that mitigate this bias.

The impact of π can be defined by the same three aspects outlined in the previous subsection: whether it influences absolute values, ratios among algorithms, or rankings among algorithms. Intuitively, one would assume algorithms that closely correlate with the logging policy have an inherent advantage, as the disadvantages inferred by the lack of interaction in offline evaluation are annulled in this setting.

3 EXPERIMENTS

The setting of our experiments is summarised in the following section. We give an overview of the recommendation algorithms we compared and go on by describing the dataset we used. Experimental results are presented and discussed in Subsections 3.3 and 3.4, following the same distinction as the research questions presented in Subsections 2.4 and 2.5 respectively.

3.1 Algorithms

Two simple baselines were used to compare algorithm performance against: a global popularity baseline (*POP*) and a sliding window popularity baseline (*POP-N*). The first sorts all items based on their number of occurrences in the full training set, and the latter sorts items based on the number of occurrences in the last N recorded interactions at the time of recommending. As the sliding window approach does not apply to the leave-one-out scheme (as it requires temporal information, which is not available), we do not include it in the LOOCV results. However, the approach proves surprisingly effective in our sliding window based evaluation method.

Apart from these baselines, we compared several well-known and widely used algorithms. The item k-nearest neighbour (*I-kNN*) algorithm computes the recommendation score $\tilde{\mathbf{R}}_{u,i}$ for a user u and an item i as a weighted sum of cosine similarities between i and items $j \in I_u$, as shown in Equation 2 [23].

$$\tilde{\mathbf{R}}_{u,i} = \frac{1}{|I_u|} \sum_{j \in I_u} \cos(\vec{i}, \vec{j}) \tag{2}$$

As a full similarity self-join on the set of items I is both time- and space-consuming, only similarities between every item and its k nearest neighbours are computed and retained. This leads to a space complexity of $O(kn)$ instead of $O(n^2)$.

The user k-nearest neighbour algorithm (*U-kNN*) is a traditional and intuitive collaborative filtering algorithm that counts the occurrences of items that u has not yet consumed among the k nearest neighbours of u. Items that are more popular with similar users are then assumed to make up better recommendations [9].

Matrix factorization algorithms explicitly compute item- and user-factors in a fixed number of latent dimensions. The recommendation score for a user u and item i is then defined as the dot-product of their latent factors. We compute the factors using the well-known Singular Value Decomposition (SVD) algorithm [31].

As the purpose of this work is not to determine which algorithm generates optimal recommendations, we refrain from investigating more advanced or recent state-of-the-art algorithms. However, it should be noted that nearest-neighbour-based algorithms have recently still been shown to attain competitive performance with the state-of-the-art [12, 30].

If the top-k recommendation list $\text{Rec}_{u,k}$ generated by any algorithm contains items that were already in the history of u, we drop them from the list and expand it. As we work with de-duplicated interactions, re-targeting is out of the scope of this paper.

The baselines, U-kNN, I-kNN and SVD were implemented using Sci- and Num-Py [14, 29]. Optimal hyper-parameters were obtained through an extensive grid search on LOOCV for optimal Recall@10 before experiments were conducted. For fair comparison, we did not recompute optimal hyper-parameters for the SW-EVAL setting or varying values of k, but retained the optimal ones for LOOCV and $k = 10$.

3.2 Dataset

Retail is a proprietary dataset obtained from the logs of a live recommender system serving a Belgian retail website, over the course of 4 months. During this period, 3 different algorithms generated recommendations in the fashion of an A/B-test. Approximately 25% of the recommendations were generated by a popularity baseline, 25% by an undisclosed algorithm, and 50% by *I-kNN*. We will respectively denote these logging policies by π_p, π_u and π_i. It is notable that the recommender system is subject to certain business rules, and top-k recommendation lists can therefore not be shown

Table 1: Characteristics of the dataset

	$\|\mathcal{D}\|$	$\|U\|$	$\|I\|$	Sparsity
Retail	734 813	189 343	10 700	99.96%

to the user as is. However, how and exactly which recommendations were effectively shown to the user is not important for the purposes of this work. Out of these 4 months, the last month acts as validation period for SW-EVAL. With 30 folds, this corresponds to daily updates and evaluations. Table 1 provides an overview of the dataset's size and properties.

3.3 LOOCV vs. SW-EVAL

In what follows we discuss the experimental results corresponding to the research question laid out in Section 2.4. To determine whether LOOCV and SW-EVAL produce comparable results as an offline evaluation procedure, we report mean Recall@k for varying k and relative performance to the best performer (in bold) for both 10-fold LOOCV and 30-fold SW-EVAL in Table 2. We set the threshold u_{min} to 100 for SW-EVAL, but generally multiple hundreds of users were included for evaluation at every fold. Results are visualised in Figure 1.

LOOCV results indicate that I-kNN is the clear best performer regardless of the value of k, exceeding the Recall of its competing algorithms with a factor of 2 and even beating the baseline algorithm with a factor of 10. However, SW-EVAL results paint an entirely different picture. First, the best performing algorithm has a mean Recall@10 that is a factor 5 smaller than reported by LOOCV. Second, we observe that the simple popularity baseline is able to attain up to 44% of the Recall@10 of the best performer, in stark contrast with the 11% reported by LOOCV. Furthermore, a simple sliding window extension boosts this further up to 71% and even 98% for $k = 5$. Third, where LOOCV concludes SVD and U-kNN to be virtually equal for $k = 10$, SW-EVAL clearly prefers SVD with 81% of the optimal performance instead of just 49%.

We observe that both for LOOCV and SW-EVAL, the gap between I-kNN and SVD closes as the number of generated recommendations grows. For the reported Recall@20, only a 3% difference remains between the two competing algorithms. The gap between U-kNN and I-kNN remains somewhat steady. A possible explanation for this is that the optimal amount of a user's neighbours to be taken into account when generating recommendations for $k = 10$ might be suboptimal for larger k, as the same candidate items might keep reappearing instead of novel recommendations. We see a similar but more stark effect for POP-N: as the optimal N obtained from the hyper-parameter optimisation procedure was rather low, the number of truly trending items might become less than k as k keeps growing. Extending POP-N to include a more advanced recency formula could certainly solve this issue.

It is clear that LOOCV and SW-EVAL do not yield comparable results for the given dataset. LOOCV is prone to overestimate model performance in general, vastly underestimate the effectiveness of popularity-based baselines, and generally rank algorithms very differently than SW-EVAL.

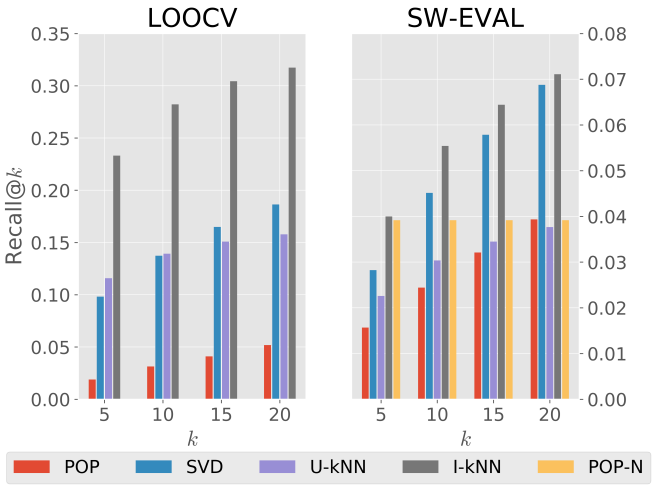

Figure 1: Mean Recall@k for the *Retail* dataset when using 10-fold LOOCV and 30-fold SW-EVAL respectively, for varying values of k. Note that the SW-EVAL y-axis is scaled down by a factor of almost 5 in comparison with the LOOCV plot.

One might note that the number of relevant items for a given user in the validation set $|\text{Rel}_u|$ is important when computing the Recall@k: for LOOCV, $|\text{Rel}_u| = 1$. If it differs greatly for SW-EVAL, lower absolute numbers are to be expected. However, we also conducted experiments where only the first item a user interacts with in the validation interval is seen as relevant, effectively setting $|\text{Rel}_u| = 1$ as well. Results were very comparable, but omitted for brevity.

3.4 Impact of Logging Policy

In what follows we discuss the experimental results corresponding to the research question laid out in section 2.5. To determine whether the logging policy π has a significant impact on results from offline evaluation procedures on $\mathcal{D}^\pi$, we report mean Recall@k for varying k, and relative performance to the best performing algorithm for both 10-fold LOOCV and 30-fold SW-EVAL on $\mathcal{D}^{\pi_u}$, $\mathcal{D}^{\pi_p}$ and $\mathcal{D}^{\pi_i}$ respectively in Table 3. Even though the number of users covered by a subset $\mathcal{D}^\pi$ of $\mathcal{D}$ will be lower than the number of users covered by the full set of transactions $\mathcal{D}$, we set u_{min} to 100 as in our previous experiments. Equivalently to those previous experiments, this lower bound was never attained. Results are visualised in Figure 2, where the top and lower row of plots respectively correspond to LOOCV and SW-EVAL results. Every column represents results on a subset of logs corresponding to a given logging policy.

When considering LOOCV results, algorithm ranking generally does not seem to be impacted by varying π or k for this specific case. However, absolute measurements as well as ratios among competing algorithms clearly are. Under the undisclosed logging policy π_u and for $k = 10$, I-kNN only achieves roughly 65% of the performance it reaches under its own logging policy π_i. When we consider the absolute performance of U-kNN over all logging

Table 2: Mean Recall@k for the *Retail* dataset when using 10-fold LOOCV and 30-fold SW-EVAL respectively, for varying values of k. Δ% denotes the relative performance of an algorithm compared to the best performer (in bold). The sliding window baseline POP-N is not included for LOOCV as it lacks the use of temporal information.

	10-fold LOOCV								30-fold SW-EVAL									
k	**POP**	Δ%	**SVD**	Δ%	**U-kNN**	Δ%	**I-kNN**	Δ%	**POP**	Δ%	**POP-N**	Δ%	**SVD**	Δ%	**U-kNN**	Δ%	**I-kNN**	Δ%
5	0.019	-92%	0.098	-58%	0.116	-50%	**0.233**		0.016	-61%	0.039	-2%	0.028	-29%	0.022	-44%	**0.040**	
10	0.031	-89%	0.137	-51%	0.139	-51%	**0.282**		0.024	-56%	0.039	-29%	0.045	-19%	0.030	-45%	**0.055**	
15	0.041	-87%	0.165	-46%	0.151	-50%	**0.304**		0.032	-50%	0.039	-39%	0.058	-10%	0.034	-46%	**0.064**	
20	0.052	-84%	0.186	-41%	0.158	-50%	**0.317**		0.039	-45%	0.039	-45%	0.069	-3%	0.038	-47%	**0.071**	

policies, we find it seems rather stable. However, the relative performance of U-kNN compared to I-kNN varies from 58% to 92% for its worst and best measurements respectively. While less pronounced, SVD exhibits similar behaviour. The popularity baseline seems moderately stable. Although less evident for π_u and π_p, a clear bias towards I-kNN is present in the data that was generated by showing I-kNN recommendations to users on the website, becoming more and more clear as k increases.

In the SW-EVAL results, the bias appears even more clear-cut. We distinctly observe that absolute values, the ratio amongst competing algorithms and the general ranking of algorithms is heavily influenced by π. For both the undisclosed algorithm π_u and the popularity policy π_p, the sliding window popularity baseline outperforms all competing algorithms for small values of k. As in the reported results from the previous section, SVD and I-kNN are able to thrive when generating more recommendations, whereas U-kNN and POP-N are much less able to do so. We suspect this is an artefact of our hyper-parameter optimisation procedure, since optimal parameters for $k = 5$ or $k = 20$ are bound to be different than those found for $k = 10$.

For the I-kNN policy π_i, a clear bias is present towards the I-kNN algorithm from the offline evaluation results. This is not surprising, as items that were effectively shown to a user have a much higher probability of being clicked than those that were not shown. Even though this clear bias is present, SVD attains up to 92% of I-kNN's performance for the largest value of k. This phenomenon is widely known as presentation bias [7], and can cause severe feedback loops if not handled properly. The effects of those feedback loops may be detrimental to the performance of a recommender system, as items in the long tail will never be considered fairly [20]. Recent related work has focused on retrieving users' intrinsic preferences when such feedback loops are present [26]. Metrics or frameworks that reward long-tail recommendations more than others are also a possible way of alleviating this issue [1, 28].

4 CONCLUSION

In this work, we have motivated the need for alternative offline evaluation procedures from LOOCV. We have identified that despite its clear flaws (ignoring all temporal information), LOOCV still remains an extremely popular technique to experimentally validate newly proposed algorithms as the state-of-the-art in recent recommender systems research [4, 8, 16–18, 21]. To this end, we have proposed a novel approach that much more tightly follows the important aspects of live recommender systems, such as their dynamic and temporal nature. We have experimentally validated on real-world data originating from the logs of a live recommender system that our approach yields very different results than LOOCV in terms of absolute metrics, ratios among competing algorithms, and mutual rankings of competing algorithms. Furthermore, LOOCV vastly underestimates the performance of advanced popularity-based approaches to recommendation.

Moreover, we motivated and discussed how live recommendation algorithms in place at the time of data collection can severely influence said data and produce a clear presentation bias, which is in turn prone to generate feedback loops. Offline evaluation results using traditional metrics on data gathered from a live recommender system are therefore heavily influenced by these biases, and lead to varying conclusions in terms of algorithm-specific performance.

4.1 Future Work

As most implicit feedback datasets for recommender systems originate from the logs of live systems, and more information about the logging policy is often unavailable, the results presented throughout this paper reveal an important issue with current evaluation and learning procedures. We intend to further validate SW-EVAL by studying the impact of the window size t_Δ, and by examining the correlation between its results and those attained through large-scale online A/B-tests for different recommendation use cases. However, as the experimental results presented in Section 3.4 show, the choice of logging policy π induces a significant bias that heavily impacts offline evaluation results on $\mathcal{D}^\pi$. To this end, a clear need rises for bias-free learning and evaluation procedures.

The work of Steck [28] tackles the issue of popularity bias by proposing "Popularity-Stratified Recall" as an improved evaluation metric. The author goes on to present a model of item popularity, and an accompanying learning procedure that optimises a matrix factorization model for said novel metric. By adapting this "Popularity-Stratified Recall" and the accompanying recommender algorithm to a "Propensity-Stratified Recall" model, one might be able to alleviate the presentation bias that is inherent in most real-world datasets.

Recent work has tackled these biases by propensity-weighting in Learning-to-Rank specifically for information retrieval systems [13]. By extending such works to include biases that are inherent to recommender systems, we believe fairer offline learning and evaluation procedures for implicit feedback recommenders with MNAR data are achievable.

Table 3: Mean Recall@k for the different A/B-groups corresponding to logging policies in the *Retail* dataset when using 10-fold LOOCV and 30-fold SW-EVAL respectively, for varying values of k. $\Delta\%$ denotes the relative performance of an algorithm compared to the best performer (in bold). The sliding window baseline POP-N is not included for LOOCV as it lacks the use of temporal information.

		10-fold LOOCV								30-fold SW-EVAL									
	k	POP	Δ%	SVD	Δ%	U-kNN	Δ%	I-kNN	Δ%	POP	Δ%	POP-N	Δ%	SVD	Δ%	U-kNN	Δ%	I-kNN	Δ%
$\mathcal{D}^{\pi_u}$	5	0.017	-89%	0.092	-40%	0.141	-8%	**0.154**		0.016	-57%	**0.037**		0.026	-28%	0.025	-31%	0.023	-36%
	10	0.029	-84%	0.127	-32%	0.171	-8%	**0.185**		0.023	-43%	0.037	-8%	**0.040**		0.035	-13%	0.032	-21%
	15	0.039	-81%	0.152	-25%	0.188	-7%	**0.204**		0.031	-37%	0.037	-26%	**0.050**		0.040	-20%	0.037	-25%
	20	0.048	-78%	0.173	-19%	0.200	-7%	**0.214**		0.037	-36%	0.037	-36%	**0.058**		0.044	-25%	0.042	-29%
$\mathcal{D}^{\pi_p}$	5	0.021	-89%	0.097	-47%	0.139	-24%	**0.183**		0.016	-58%	**0.037**		0.029	-21%	0.230	-38%	0.032	-14%
	10	0.033	-85%	0.134	-41%	0.169	-25%	**0.225**		0.025	-47%	0.037	-23%	**0.048**		0.031	-34%	0.046	-4%
	15	0.045	-81%	0.159	-35%	0.186	-24%	**0.245**		0.033	-47%	0.037	-40%	**0.061**		0.038	-39%	0.053	-14%
	20	0.056	-78%	0.180	-30%	0.197	-23%	**0.257**		0.039	-45%	0.037	-48%	**0.071**		0.041	-42%	0.058	-18%
$\mathcal{D}^{\pi_i}$	5	0.018	-91%	0.102	-52%	0.137	-35%	**0.212**		0.016	-61%	0.038	-6%	0.027	-33%	0.027	-35%	**0.041**	
	10	0.031	-89%	0.142	-50%	0.165	-42%	**0.284**		0.024	-58%	0.038	-33%	0.045	-21%	0.034	-40%	**0.057**	
	15	0.042	-86%	0.170	-44%	0.180	-41%	**0.406**		0.032	-51%	0.038	-43%	0.057	-15%	0.039	-42%	**0.067**	
	20	0.051	-84%	0.192	-40%	0.189	-41%	**0.319**		0.040	-46%	0.038	-48%	0.068	-8%	0.042	-43%	**0.074**	

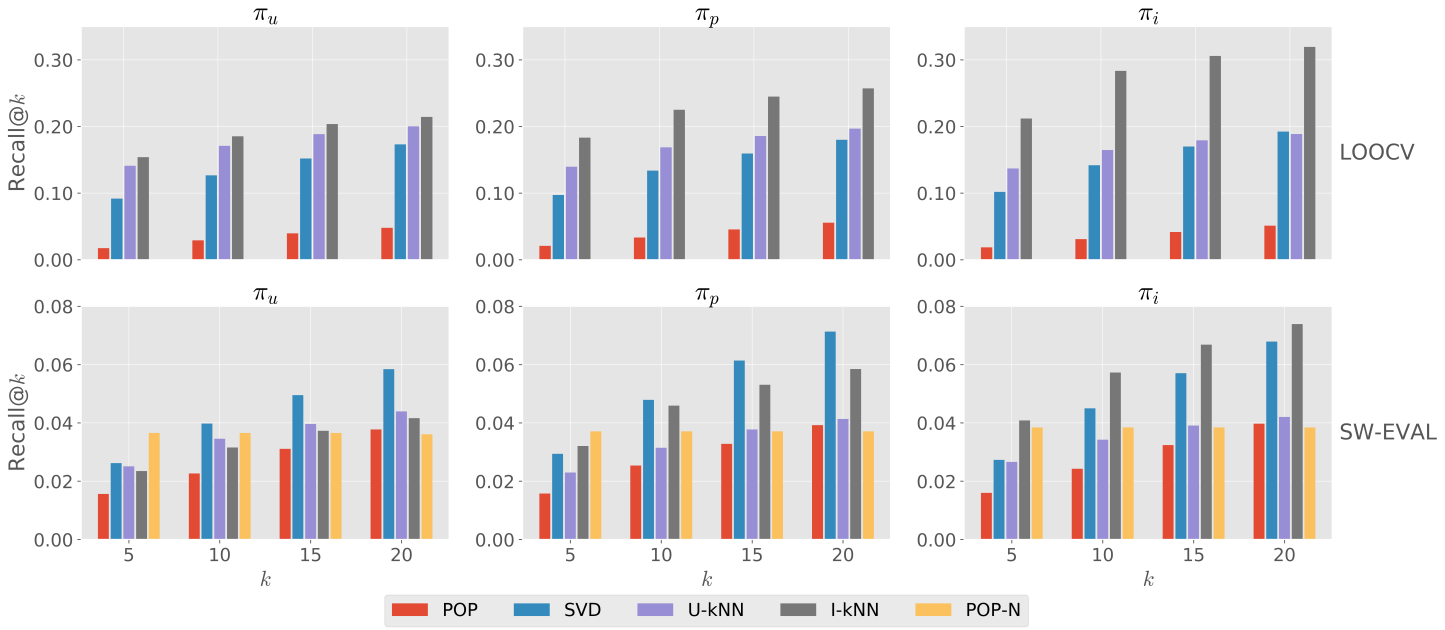

Figure 2: Mean Recall@k for the *Retail* dataset when using 10-fold LOOCV and 30-fold SW-EVAL respectively, subdivided by logging policy, for varying values of k. Note that the SW-EVAL y-axes are scaled down by a factor of almost 5 in comparison with the LOOCV plots.

REFERENCES

[1] H. Abdollahpouri, R. Burke, and B. Mobasher. 2017. Controlling Popularity Bias in Learning-to-Rank Recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys '17)*. ACM, 42–46.

[2] A. Agarwal, S. Basu, T. Schnabel, and T. Joachims. 2017. Effective Evaluation Using Logged Bandit Feedback from Multiple Loggers. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM, 687–696.

[3] J. Beel, M. Genzmehr, S. Langer, A. Nürnberger, and B. Gipp. 2013. A Comparative Analysis of Offline and Online Evaluations and Discussion of Research Paper Recommender System Evaluation. In *Proc. of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys '13)*. 7–14.

[4] E. Christakopoulou and G. Karypis. 2016. Local Item-Item Models For Top-N Recommendation. In *Proc. of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 67–74.

[5] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber. 2014. Offline and Online Evaluation of News Recommender Systems at Swissinfo.Ch. In *Proc. of the 8th ACM Conference on Recommender Systems (RecSys '14)*. 169–176.

[6] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham, and S. Dollé. 2018. Offline A/B Testing for Recommender Systems. In *Proc. of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. ACM, 198–206.

[7] C. A. Gomez-Uribe and N. Hunt. 2015. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management of Information Systems* 6, 4, Article 13 (Dec. 2015), 19 pages.

[8] R. He, W. Kang, and J. McAuley. 2017. Translation-based Recommendation. In *Proc. of the 11th ACM Conference on Recommender Systems (RecSys '17)*. ACM, 161–169.

[9] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*. ACM, 230–237.

[10] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.

[11] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proc. of the 8th IEEE International Conference on Data Mining (ICDM '08)*. 263–272.

[12] D. Jannach and M. Ludewig. 2017. When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation. In *Proc. of the 11th ACM Conference on Recommender Systems (RecSys '17)*. ACM, 306–310.

[13] T. Joachims, A. Swaminathan, and T. Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proc. of the 10th ACM International Conference on Web Search and Data Mining (WSDM '17)*. ACM, 781–789.

[14] E. Jones, T. Oliphant, and P. Peterson. 2014. SciPy: open source scientific tools for Python. (2014).

[15] R. Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. of the 4th International Joint Conference on Artificial Intelligence (IJCAI '95)*, Vol. 14. 1137–1145.

[16] X. Ning and G. Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *Proc. of the 11th IEEE International Conference on Data Mining (ICDM '11)*. 497–506.

[17] Y. Ning, Y. Shi, L. Hong, H. Rangwala, and N. Ramakrishnan. 2017. A Gradient-based Adaptive Learning Framework for Efficient Personal Recommendation. In *Proc. of the 11th ACM Conference on Recommender Systems (RecSys '17)*. ACM, 23–31.

[18] R. Otunba, R. A. Rufai, and J. Lin. 2017. MPR: Multi-Objective Pairwise Ranking. In *Proc. of the 11th ACM Conference on Recommender Systems (RecSys '17)*. ACM, 170–178.

[19] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. 2008. One-Class Collaborative Filtering. In *Proc. of the 8th IEEE International Conference on Data Mining (ICDM '08)*. 502–511.

[20] Y. Park and A. Tuzhilin. 2008. The Long Tail of Recommender Systems and How to Leverage It. In *Proc. of the 1st ACM Conference on Recommender Systems (RecSys '08)*. ACM, 11–18.

[21] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proc. of the 25th Conference on Uncertainty in Artificial Intelligence (UAI '09)*. AUAI Press, 452–461.

[22] M. Rossetti, F. Stella, and M. Zanker. 2016. Contrasting Offline and Online Results when Evaluating Recommendation Algorithms. In *Proc. of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 31–34.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proc. of the 10th International Conference on World Wide Web (WWW '01)*. ACM, New York, NY, USA, 285–295.

[24] A. Schein, A. Popescul, L. Ungar, and D. Pennock. 2002. Methods and Metrics for Cold-start Recommendations. In *Proc. of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*. ACM, 253–260.

[25] G. Shani and A. Gunawardana. 2011. Evaluating Recommendation Systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, 257–297.

[26] A. Sinha, D. Gleich, and K. Ramani. 2016. Deconvolving Feedback Loops in Recommender Systems. In *Advances in Neural Information Processing Systems*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Vol. 29. Curran Associates, Inc., 3243–3251.

[27] H. Steck. 2010. Training and Testing of Recommender Systems on Data Missing Not at Random. In *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*. ACM, 713–722.

[28] H. Steck. 2011. Item Popularity and Recommendation Accuracy. In *Proc. of the 5th ACM Conference on Recommender Systems (RecSys '11)*. ACM, 125–132.

[29] S. Van Der Walt, S. Colbert, and G. Varoquaux. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* 13, 2 (2011), 22–30.

[30] K. Verstrepen and B. Goethals. 2014. Unifying Nearest Neighbors Collaborative Filtering. In *Proc. of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, 177–184.

[31] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman. 2005. Using singular value decomposition approximation for collaborative filtering. In *Proc. of the 7th IEEE International Conference on E-Commerce Technology (CEC '05)*. 257–264.