Characterization of Fair Experiments for Recommender System Evaluation – A Formal Analysis

Pablo Castells
Universidad Autónoma de Madrid
pablo.castells@uam.es

Rocío Cañamares
Universidad Autónoma de Madrid
rocio.cannamares@uam.es

ABSTRACT

Recommender system experiments for evaluation involve a considerable number of configuration settings and decisions. Not only has this been a breeding ground for considerable methodological divergence in experimental practice the field, but it raises que issue to what extent different experimental approaches might be equally reliable. In this paper we address the question on the side of experimental fairness, meaning to what extent the outcome of a comparative experiment matches the underlying truth, and is not biased towards favoring a priori a particular algorithmic approach. Upon an abstract unified characterization of experiment configurations, we identify a formal condition for an experiment to be fair and informative. Based on this, we examine common and particular evaluation procedures described in the literature. We show that common offline approaches are subject to strong biases that may compromise the reliability of comparative evaluation results, while we identify alternatives that ensure fairness. We confirm and illustrate some of our theoretical findings in an experiment with partially synthetic data.

KEYWORDS

recommender systems; offline and online evaluation; experimental design; metrics; bias; non-random missing data

1 INTRODUCTION

Recommender system experiments for evaluation involve a considerable number of configuration settings and decisions [11,23], more than may be often explicitly reported or even settled for in full awareness by the experimenters. Moreover, the field has adapted evaluation procedures from adjacent disciplines (such as machine learning and information retrieval), and the recommendation task has particularities of its own that borrowed methodologies did not necessarily have to care for in the tradition they were adapted from [1]. The degree of divergence in experimental practice is being indeed often acknowledged as a hindrance to progress in the recommender systems field [16]. The divergence goes beyond the peculiar requirements of specific recommendation domains and tasks, and as such should be avoidable to a large extent. The lack of a clear common terminology and conceptualization for certain experimental details adds to the difficulty.

Recent research has furthermore found that recommender system evaluation is vulnerable to biases and missing not at random (MNAR) conditions in the data [19,20,21,25], in such a way that the qualitative outcome of an experiment might in some cases disagree with the true situation as to what system is being the most effective in satisfying user needs [4]. Even though important progress has been achieved in confirming, describing, avoiding or coping with specific experimental distortions [12,15,21,25], we find that it is possible to reach a deeper and wider understanding at a more fundamental level, from a general methodological perspective.

In this paper we seek progress in this direction by, first, laying a basis that helps characterize and reason about the key elements that make up an experimental setting for recommender system evaluation in formal and unambiguous ways. We seek a concise unified conceptualization that is abstract enough to be formally analyzed and reasoned upon, but sufficiently specific to match the elements and requirements of common experimental practice in the field.

Building upon this, we seek to identify, state and formalize the fundamental conditions for experimental outcomes to be reliable and representative of the target qualities that evaluation aims to capture: the conditions that ensure an evaluation is fair to the compared alternatives, and not biased towards favoring a priori any particular algorithmic approach. Using this characterization, we examine common and alternative evaluation procedures in the literature, and determine whether they grant the fairness condition or not, why and to what extent.

In our analysis we find that –with a particular non-fully satisfactory exception– it is impossible to avoid biases in evaluation with common public datasets collected from natural user activity –at least with any processing approach reported in the literature, to the best of our knowledge. We show that fair offline evaluation is nonetheless possible by appropriately collecting user input through randomized user preference samples. We further confirm that online AB testing, with its known cost and reproducibility tradeoffs, may ensure fair comparative evaluation as well.

2 EXPERIMENT DESIGN CHARACTERIZATION

At an abstract level, we may consider an evaluation experiment involves two essential actors: a recommender system (or rather, a set of systems), and an experimenter. The former can be seen by the latter as a black box ready to take some data as input (involving users and items) and return a ranking of items for each user. The experimenter collects a set of data, subdivides it as appropriate for the experiment, supplies input to the system, and computes metrics on the returned output. The experimenter has partial knowledge of all the existing information (interaction and features) involving users and items –she knows as much as she could collect for the experiment. The system on its side has access to yet a subset of the experimenter's knowledge that the latter decides to show (the training data).

In this general mind frame, given an experiment involving a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$, we can describe an experiment configuration in terms of five fundamental subsets of $\mathcal{U} \times \mathcal{I}$, illustrated in Figure 1. Note that it would also be possible to develop all the definitions that follow, and the forthcoming analysis, in terms of subsets of items for a given fixed target user, instead of sets of user-item pairs. We find nonetheless that taking $\mathcal{U} \times \mathcal{I}$ as the domain of discourse may induce wider insights and is more flexible for further elaborations beyond our present work**.** We shall thus formally describe an experiment configuration by a tuple $\mathcal{C} = \langle \mathcal{R}, \mathcal{J}_{train}, \mathcal{J}_{test}, R, \mathcal{T} \rangle$ consisting of the sets that we define next.

Relevance $\mathcal{R} \subset \mathcal{U} \times \mathcal{I}$. We have $(u, i) \in \mathcal{R}$ if the user u likes the item i. Relevance is in general partially observed: even the user is generally not aware of all the items she would appreciate –she would not be using a recommender system otherwise.

Judgments $\mathcal{J} \subset \mathcal{U} \times \mathcal{I}$ represent partial knowledge about $\mathcal{R}$, in normal scenarios where not all user preferences are known to the experimenter, but only a sample thereof. This sample can consist of ratings or implicit feedback records, though we shall use the term "judgment" for a more general and flexible purpose, as we shall see. We consider the judgment set is made of two subsets: $\mathcal{J} = \mathcal{J}_{train} \cup \mathcal{J}_{test}$. The training subset $\mathcal{J}_{train}$ includes user-item pairs for which the system is supplied input observations (about relevance) for the user: and the test subset $\mathcal{J}_{test}$ contains relevance judgments the experimenter uses to compute evaluation metrics on the recommendation. Unlike $\mathcal{R}$, the judgment set $\mathcal{J}$ and its subsets do not themselves represent knowledge, but the domain of some available knowledge. That is, $\mathcal{J}$ is a set of user-item pairs (a sample) for which binary relevance knowledge is available in the experiment.

Recommendation $R \subset \mathcal{U} \times \mathcal{I}$. This represents the output of the system under evaluation, where $(u, i) \in R$ if the system recommends the item i to user u. Even though a recommendation is an ordered set, our formal analysis shall focus on set-oriented metrics such as P@k and Recall@k, for the purpose of which we would simply consider that R contains the top k items ranked by the recommendation algorithm for each user in the system. The generalization of our theoretical analysis to metrics with a finer rank sensitivity, such as nDCG, can be shown empirically, but is considerably challenging to handle analytically.

Targets $\mathcal{T} \subset \mathcal{U} \times \mathcal{I}$. This defines a set of user-item pairs to which we may require the recommendations be restricted in the experiment. The target set is sometimes a way to simplify an experiment and/or reduce its size [1,8]. In exchange for potential convenience, selecting targets alters the original task, inasmuch as it excludes candidate outputs that the recommender system would have to deal with in a real situation.

The above described sets are to a large extent sufficient to characterize the key data and design of an experiment. We shall in fact fully describe in sections 5 and 6 a set of common experimental protocols just in terms of how such sets are defined and sampled. We will additionally consider the following set of assumptions, which might not be strictly necessary, but hold in essentially all the experimental approaches reported in the literature as far as our knowledge and practical imagination goes. New experimental settings (or formulations thereof) beyond some of these assumptions might nonetheless be conceived.

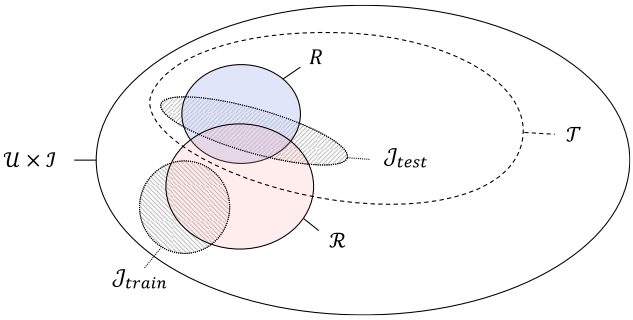

Figure 1: Illustration of the fundamental sets involved in a recommender system evaluation experiment.

- $\mathcal{J}_{train} \cap R = \emptyset$. In our present research we focus on recommendation tasks that avoid repeated item consumption, which are typically the ones where recommendation brings the most value. We hence consider the system takes care of excluding from its output recommendation R any user-item pairs in $\mathcal{J}_{train}$ (for which input data were supplied to the system).
- $\mathcal{J}_{train} \cap \mathcal{J}_{test} = \emptyset$. Since evaluation metrics apply to $\mathcal{J}_{test} \cap R$, and we are assuming $\mathcal{J}_{train} \cap R = \emptyset$, we may also assume $\mathcal{J}_{train}$ and $\mathcal{J}_{test}$ are disjoint sets.
- $R \subset \mathcal{T}$ by definition of $\mathcal{T}$, and $\mathcal{J}_{train} \cap \mathcal{T} = \emptyset$ since recommendations should not overlap with training judgments. The largest possible target set is thus $\mathcal{T} = (\mathcal{U} \times \mathcal{I}) \setminus \mathcal{J}_{train}$, and can be assumed as the default option unless stated otherwise.
- $\mathcal{J}_{test} \subset \mathcal{T}$. It would make little sense in general to waste test data by excluding it from candidate recommendations. If an experiment aims to focus on users or items with some specific property (e.g. recommendation of long-tail items), we may then still take this condition for granted by removing the "uninteresting" targets from both $\mathcal{J}_{test}$ and $\mathcal{T}$.

3 ACCURACY EVALUATION: THEORETICAL VALUE AND APPROXIMATION

Many different qualities can be involved in defining what a good recommendation is [6,23]. In our present analysis we focus primarily on relevance as one fundamental core requirement for recommendations to make sense. We start by formalizing basic set-oriented accuracy metrics in terms of configuration sets defined in the previous section: $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{T}$. We shall then formulate the difference between the true theoretical value of metrics, and the estimates than can be computed in an experiment with a limited sample $\mathcal{J}_{test}$ of the full relevance knowledge $\mathcal{R}$.

3.1 Theoretical Metric Formulation

From the relevance point of view, the goal of a good accurate recommendation is, broadly speaking, to maximize the set $R \cap \mathcal{R}$ of relevant recommendations. We now just need a good metric and a good experimental protocol to measure as faithfully as possible, and under the suitable angle, how large or dense is this set. Let us consider for instance precision and recall as simple metrics, which in our notation correspond to:

$$P = \frac{|R \cap \mathcal{R}|}{|R|} \qquad Recall = \frac{|R \cap \mathcal{R}|}{|\mathcal{R}|}$$

If we consider relevance $\mathcal{R}$ and recommendation R as binary random variables in $\mathcal{U} \times \mathcal{I}$, precision can be read as the probability that a recommended item is relevant for the target user, and recall as the probability that an item the user likes is recommended:

$$P = p(\mathcal{R}|R) \qquad Recall = p(R|\mathcal{R})$$

3.2 Empirical Metric Estimates

The above metric definitions cannot be exactly computed in general situations where the experimenter knowledge of $\mathcal{R}$ is partial: the experimenter has knowledge for a test sample $\mathcal{J}_{test}$, based on which she can only compute metric estimates for the respective theoretical definitions, namely:

$$\hat{P} = \frac{|R \cap \mathcal{R} \cap \mathcal{J}_{test}|}{|R|} = p(\mathcal{R}, \mathcal{J}_{test}|R)$$

$$\hat{R}ecall = \frac{|R \cap \mathcal{R} \cap \mathcal{J}_{test}|}{|\mathcal{R} \cap \mathcal{J}_{test}|} = p(R|\mathcal{J}_{test}, \mathcal{R})$$

If we had full oracle knowledge, that is, as the test sample expands to the full data space, we would just get $\hat{P} \to P$ and $\hat{R}ecall \to Recall$. In the natural case where the sample is comparatively very small $|\mathcal{J}_{test}| \ll |(\mathcal{U} \times \mathcal{I}) \setminus \mathcal{J}_{train}|$, the estimation can considerably diverge from the true metric value. The magnitude deviations in the metrics are not important as long as comparisons between systems are preserved. However this will not be the case if $\mathcal{J}_{test}$ is not just small, but biased in certain ways, as we shall analyze. We start by identifying and postulating sufficient conditions that guarantee system comparison preservation in metric estimates.

4 INFORMATIVE EXPERIMENT

In terms of the distinction between metric values and estimates, and the sets $\mathcal{U}$, $\mathcal{I}$, $\mathcal{J}_{train}$ and $\mathcal{T}$, we can state and formalize what it means for an experiment to be informative with respect to the real setting in which the algorithms under evaluation are to be deployed.

4.1 General Conditions

In defining what we expect from an experiment to be informative, in order to make a decision and/or a choice over a set of algorithmic (or algorithm configuration) alternatives in production systems, we identify two desirable conditions for an experimental design.

4.1.1 Metric estimate fairness. Even if the metric estimates on sample data can drastically diverge from the true metric values we would compute with full oracle knowledge, we need the estimates to at least preserve qualitative system comparisons, if they are to inform of the real effectiveness of the evaluated algorithms. This means $\hat{M}(R) \propto M(R)$ where M is the chosen evaluation metric. If the estimates furthermore approximate the quantitative metric values, we have $\hat{M}(R) \sim M(R)$.

4.1.2 Representativeness. An experiment should be a reasonable (simplified or reduced) match of the real task where we are to make a decision. This means the experiment configuration $\mathcal{C}$ should represent as fair an example as possible of what the evaluated algorithm would have to achieve in the real setting. To begin with, the sets $\mathcal{U}$, $\mathcal{I}$, and $\mathcal{J}_{train}$ should fairly represent (the characteristics of) the data a recommender is fed while in production. On the other hand, the selection of the target set $\mathcal{T}$ should likewise mimic a real situation –in the absence of any particular specification from that side, the target should cover the largest possible set, i.e. $\mathcal{T} = (\mathcal{U} \times \mathcal{I}) \setminus \mathcal{J}_{train}$.

There are further conditions one may wish an experiment to meet, such as reproducibility [16], scale, etc. In our present study we focus on the above two requirements as fundamental properties. We have stated the representativeness condition at a rather general level; we shall leave it at that point here, and we will focus on further developing the estimate fairness condition. In particular, we shall seek to postulate sufficient conditions for an experiment to meet this property. In the section following that, we shall examine common evaluation protocols reported in the literature, and analyze to what extent they meet or not the two informativeness conditions posited here.

4.2 Fair Metric Estimates

In order to identify fairness conditions, we can formally relate metrics and their estimates in terms of their probabilistic expression. By straightforward application of Bayes' rule, we have:

$$\hat{P} = p(\mathcal{J}_{test}|\mathcal{R}, R)\ P$$

For the metric estimates to be fair, we would just need the multiplying term in the above relations to be a constant for all recommendations. Thus, we need $p(\mathcal{J}_{test}|\mathcal{R}, R)$ to be the same for all the recommendations R to be compared with $\hat{P}$, that is: $p(\mathcal{J}_{test}|\mathcal{R}, R) \sim p(\mathcal{J}_{test}|\mathcal{R}, \mathcal{T})$. In other words, in a fair experiment $\mathcal{J}_{test}$ should be conditionally independent from R given $\mathcal{R}$. It is very easy to see that this is equivalent to the test sample being uniform (to be more precise, identically distributed) over relevant target items:

$$\forall i \in \mathcal{I}, p(\mathcal{J}_{test}|\mathcal{R}, \mathcal{T}, i) \sim p(\mathcal{J}_{test}|\mathcal{R}, \mathcal{T}) \tag{1}$$

We reach an equivalent finding for recall; the approximate estimate is related to the theoretical value by:

$$\hat{R}ecall = \frac{p(\mathcal{J}_{test}|\mathcal{R}, R)}{p(\mathcal{J}_{test}|\mathcal{R})}\ Recall$$

Hence we get fair estimates of recall by the same conditions as for precision.

5 JUDGMENT SAMPLING: ANALYSIS OF EVALUATION PROTOCOLS

We now examine state of the art experimental protocols in light of the analysis of the previous section. According to how the training and test judgments are sampled, we broadly distinguish three general settings:

1. Free offline user feedback.
2. Forced offline user judgments.
3. Online AB test.

We shall now subdivide these approaches into variants and examine them in the next subsections, according to different alternatives for sampling the training and test subsets that each variant enables. We shall assume for now the largest possible target set $\mathcal{T} \leftarrow (\mathcal{U} \times \mathcal{I}) \setminus \mathcal{J}_{train}$ as the default option. Later in section 6 we shall comment on the effects of particular target sampling approaches in offline evaluation.

5.1 Free Offline User Feedback

Users leave (implicit or explicit) traces of their preference for items in the spontaneous use of an application. The evidenced preference of users for items corresponds to the judgment set $\mathcal{J}$ in out representation, and is typically referred to as *ratings* or *implicit feedback* in the recommender systems literature. The judgments are collected in this process and made available for offline evaluation. The judgment set is split into $\mathcal{J}_{train}$ and $\mathcal{J}_{test}$ for metric computation. Common public datasets, such as MovieLens [10], Netflix (https://www.netflixprize.com), Celma's Last.fm samples [7], Million Song [2], etc., are examples of this data sampling kind.

Different judgment splitting approaches are possible in this setting. We can identify the following options as most common in the literature (namely, 1 and 4 below), or having been explicitly proposed to cope with experimental biases (2 and 3):

1. Random uniform split [11,23].
2. Flat test over items [1].
3. Popularity percentile partition [1].
4. Temporal split [17,18].

We describe and analyze next each of the above split protocols in terms of informativeness as defined in section 4, showing that all of them are potentially biased and unfair in their own –stronger or weaker– way. Some incur moreover on representativeness limitations in exchange for bias reduction or practical simplicity.

5.1.1 Random uniform rating split. Judgments are sampled from $\mathcal{J}$ into $\mathcal{J}_{train}$ and $\mathcal{J}_{test}$ uniformly at random, based on some given Bernoulli probability (the *split* ratio) [11,23]. For every judgment in $\mathcal{J}$, a weighted coin (the same for all judgments) is flipped to decide whether the judgment is placed in $\mathcal{J}_{train}$ or $\mathcal{J}_{test}$. The split is hence entirely independent from any property of users or items (such as relevance), that is $p(\mathcal{J}_{test}|\mathcal{J},\mathcal{A},u,i) = p(\mathcal{J}_{test}|\mathcal{J})$ whatever $\mathcal{A}$ might be, and same for training data.

$\mathcal{J}_{test}$ is however not necessarily independent from R, inasmuch as it is not independent from the recommender's input $\mathcal{J}_{train}$ – quite the contrary, it is drawn from the same distribution: the one from which $\mathcal{J}$ (user interaction) is sampled. The test sample therefore gets the same, usually quite strong biases as are present in the internal and external processes through which users come to freely interact with items and produce judgments. To see this formally, taking $\mathcal{T} \leftarrow \mathcal{I} \setminus \mathcal{J}_{train}$ as a default setting, we can write:

$$p(\mathcal{J}_{test}|\mathcal{R},\mathcal{T},i) = p(\mathcal{J}_{test}|\mathcal{R},\neg\mathcal{J}_{train},i) = \frac{p(\mathcal{J}_{test}|\mathcal{R},i)}{1-p(\mathcal{J}_{train}|\mathcal{R},i)} \quad (2)$$

Now since $p(\mathcal{J}_{test}) = p(\mathcal{J}_{test}|\mathcal{J})\,p(\mathcal{J})$, and the same for training, with the aforementioned independence conditions we have:

$$p(\mathcal{J}_{test}|\mathcal{R},\mathcal{T},i) = \frac{p(\mathcal{J}_{test}|\mathcal{J})\,p(\mathcal{J}|\mathcal{R},i)}{1-p(\mathcal{J}_{train}|\mathcal{J})\,p(\mathcal{J}|\mathcal{R},i)} \propto p(\mathcal{J}|\mathcal{R},i) \quad (3)$$

According to equation 1, precision would be unbiased only if $p(\mathcal{J}|\mathcal{R},i) \sim p(\mathcal{J}|\mathcal{R})$ would not depend on i, that is, if user ratings were conditionally independent from the item given relevance. It is easy to see that this is equivalent to the number of positive training ratings of each item being just proportional to the number of users who liked the item. This is not necessarily the case: two items may please the same number of people, yet one may have received more user interaction for some reason (e.g. by having entered earlier in the system, or having been more advertised). In general, the evaluation is unfairly biased towards favoring systems that recommend relevant items with many ratings, that is, popular items [4].

Through the popularity bias, the experiments can be indirectly biased to any user or item feature that popularity is dependent on. For instance, if users tend to engage with recently released items more often than old ones (for reasons other than taste), an algorithm that recommends new items would be unfairly favored by the experiment.

5.1.2 Flat test over items. Proposed with the purpose of mitigating popularity biases [1], in this protocol all items get the same number t of test judgments, sampled uniformly at random from $\mathcal{J}$ (again, independently from any particular item or user feature), which means $p(\mathcal{J}_{test}|i) \sim t/|\mathcal{U}|$ for all i. With this split protocol, equation 2 leads to:

$$p(\mathcal{J}_{test}|\mathcal{R},\mathcal{T},i) = \frac{p(\mathcal{J}|\mathcal{R},i)\;p(\mathcal{J}_{test}|i)/p(\mathcal{J}|i)}{1-p(\mathcal{J}|\mathcal{R},i)\;(1-p(\mathcal{J}_{test}|i)/p(\mathcal{J}|i))} \propto \frac{1}{p(\mathcal{J}|i)(1/p(\mathcal{J}|\mathcal{R},i)-1)} \quad (4)$$

We thus find that the test sample distribution has a complex functional relation to the global judgment distribution and the judgment distribution over relevant items, from which complex biases can arise. For instance, if the conditional judgment probability given relevance was the same for all items, it is easy to see that $p(\mathcal{J}_{test}|\mathcal{R},\mathcal{T},i) \propto 1/p(\mathcal{J}|i)$, whereby the experiment would undervalue the recommendation of frequently rated items –a bias against popularity. On the other hand, if judgments were distributed independently from the relevance of items, then we would get just the opposite bias $p(\mathcal{J}_{test}|\mathcal{R},\mathcal{T},i) \propto p(\mathcal{J}|i)$. Hence the evaluation fairness depends on the strength or weakness of the judgment probability with respect to user tastes, and the biases can point one way or the opposite depending on the particular balance between the probabilistic drifts.

5.1.3 Popularity percentile partition. As an alternative way to reduce popularity biases [1], the item set in this approach is partitioned into m popularity bins $\mathcal{I} = \bigcup_{k=1}^{m} \mathcal{I}_k$, with $i \in \mathcal{I}_k \wedge j \in \mathcal{I}_{k+1} \Rightarrow p(\mathcal{J}_{train}|\mathcal{R},i) \geq p(\mathcal{J}_{train}|\mathcal{R},j)$, and approximately equal number of judgments per bin $\sum_{i\in\mathcal{I}_k} p(\mathcal{J}_{train}|\mathcal{R},i) \sim p(\mathcal{J}_{train}|\mathcal{R})/m$. [1] The evaluation is broken down into m experiments, the outcome of which is averaged to a final metric value. Each experiment takes $\mathcal{J}_{train} \cap (\mathcal{U} \times \mathcal{I}_k)$ and $\mathcal{J}_{test} \cap (\mathcal{U} \times \mathcal{I}_k)$ as training and test data, where e.g. a random split may have been applied. Each experiment would hence display the popularity corresponding to the split procedure, but $\mathcal{J}_{train}$ has a closer to uniform distribution over $\mathcal{U} \times \mathcal{I}_k$ than over $\mathcal{U} \times \mathcal{I}$, i.e. the variance of popularity is smaller when broken down into percentiles, hence $p(\mathcal{J}_{train}|\mathcal{I}_k,\mathcal{R},i)$ is closer to $p(\mathcal{J}_{train}|\mathcal{I}_k,\mathcal{R})$ than in the previous settings, whereby this may procure a weaker popularity bias and a better approximation to the theoretical metric values.

[1] The original approach [1] is defined in terms on $p(\mathcal{J}|i)$ instead of $p(\mathcal{J}_{train}|\mathcal{R},i)$, and percentiles (by number of items) instead of bins (by number of judgments). Though roughly equivalent in practice, we can now understand the approach defined here is more rigorous and better serves its original purpose.

On the downside, the recommender systems are fed pieces $\mathcal{J}^k_{train}$ of training data as input, instead of the whole training set $\mathcal{J}_{train}$. The task is thus modified: we are evaluating how well on average the algorithms are able to recommend items within a given popularity stratum, which may compromise the representativeness of the experiment with regards to the real task.

5.1.4 Temporal rating split. In this setting $\mathcal{J}_{train}$ contains all ratings entered up to some point in time, and $\mathcal{J}_{test}$ contains the data collected after that point [17,18]. In the previous protocols we have seen that popularity biases arise from functional dependencies between the test and training samples and the user feedback distribution. In fact, as far as the test sample is independent from the input of the evaluated systems (and $\mathcal{J}_{train}$ is an essential part of this input), it should be independent from their output as well, as required for fairness.

In a temporal split, $\mathcal{J}_{train}$ and $\mathcal{J}_{test}$ are produced by the same processes, determined by the same variables, only at different time periods. To the extent that the distribution of such variables and processes does not change completely over time, we may certainly expect dependencies between the test and training samples, manifested as a similar popularity (and/or feature-related) biases as in settings a) and b). Inasmuch as the distributions experience some degree of change, the dependence would become weaker than in a random split, where the training and test distributions are sampled from an identical rating distribution. We may not expect however the setting to be bias-free in general.

A temporal split enhances on the other hand representativeness with respect to the previous time-unaware split procedures. It better mimics a real setting where the system's task is to predict the future based on the past.

5.2 Forced Offline User Judgments

An alternative to recording user feedback in a natural interaction setting is to collect test judgments $\mathcal{J}_{test}$ uniformly at random at the request of the experimenter. Examples of this kind include a release of data from Yahoo! [20], and the CM100k dataset [4,5]. Since users are explicitly prompted for test feedback, they can be asked, in addition to their taste, whether or not they were familiar with the item before being surveyed. This information was collected for instance in CM100k. As to the training data, it can be obtained in essentially two different ways: a) by free user feedback [20], just as in the datasets characterized in the previous section, and b) by forced uniform sampling [4], similarly to the test judgments. We analyze next the implications of each option.

5.2.1 Free training judgments. Items $\mathcal{I}$ and users $\mathcal{U}$ are sampled uniformly at random from some larger user-item space, and $\bar{\mathcal{J}}_{test}$ is sampled uniformly at random within $\mathcal{U} \times \mathcal{I}$. Then we sample $\bar{\mathcal{J}}_{train} \subset \mathcal{U} \times \mathcal{I}$ by spontaneous user interaction in the user-item subset at hand. The training data $\bar{\mathcal{J}}_{train}$ can simply consist of all the available observations in the system involving the sample $\mathcal{U} \times \mathcal{I}$. The aforementioned Yahoo! dataset [20] is an example of this kind, although users and items were not sampled from the entire dataset, but from a filtered subset meeting some minimum requirement in the number of interaction records.

In this protocol $\bar{\mathcal{J}}_{test}$ and $\bar{\mathcal{J}}_{train}$ are not guaranteed to be disjoint, hence a decision needs to be made as to where to place their intersection, i.e. whether to remove it from the training data or the test sample, since the training and test sets need to be disjoint. We show next that the latter is the correct choice.

Removing the overlap from the test set means taking $\mathcal{J}_{test} \leftarrow \bar{\mathcal{J}}_{test} \setminus \bar{\mathcal{J}}_{train}$, and $\mathcal{J}_{train} \leftarrow \bar{\mathcal{J}}_{train}$. Since $\bar{\mathcal{J}}_{test}$ was uniformly distributed over $\mathcal{U} \times \mathcal{I}$, so is $\mathcal{J}_{test} = \bar{\mathcal{J}}_{test} \setminus \bar{\mathcal{J}}_{train}$ over $(\mathcal{U} \times \mathcal{I}) \setminus \bar{\mathcal{J}}_{train} = (\mathcal{U} \times \mathcal{I}) \setminus \mathcal{J}_{train}$. The estimates of precision and recall are unbiased given these conditions. This is therefore the ideal setting in fairness (unbiased for all purposes) and representativeness (training data are collected in the same way as in a production environment). Unfortunately no currently publicly available dataset is built exactly this way, as far as our knowledge goes.

On the other hand, removing the overlap from the training set means taking $\mathcal{J}_{train} \leftarrow \bar{\mathcal{J}}_{train} \setminus \bar{\mathcal{J}}_{test}$, and $\mathcal{J}_{test} \leftarrow \bar{\mathcal{J}}_{test}$. The representativeness is slightly hurt since the training data $\bar{\mathcal{J}}_{train}$ that a real system would get as input is altered. Furthermore, we have:

$$p(\mathcal{J}_{test}|\mathcal{R}, \neg\mathcal{J}_{train}, i) = \frac{p(\bar{\mathcal{J}}_{test})}{1 - p(\bar{\mathcal{J}}_{train}|\mathcal{R}, i)\, p(\neg\bar{\mathcal{J}}_{test})} \propto p(\bar{\mathcal{J}}_{train}|\mathcal{R}, i)$$

where the first step holds because $R \cap \mathcal{J}_{train} = \emptyset$, and the second step applies $\mathcal{J}_{train} = \bar{\mathcal{J}}_{train} \setminus \bar{\mathcal{J}}_{test}$ and the fact that $\bar{\mathcal{J}}_{test}$ is uniform and hence independent from any variable in all of $\mathcal{U} \times \mathcal{I}$. We hence get a bias towards favoring popular recommendations.

From the description of the Yahoo! dataset [20], we may infer the overlap was removed from the training set, whereby it should suffer from the bias. If the test sample is however very sparse (i.e. $p(\bar{\mathcal{J}}_{test}) \ll 1$) then $p(\mathcal{J}_{test}|\mathcal{R}, i)$ is a monotonically increasing but quite flat function of $p(\bar{\mathcal{J}}_{train}|\mathcal{R}, i)$ in the equation above, and the bias can be quite weak, possibly hardly even noticeable. The loss in representativeness can be similarly negligible.

5.2.2 All judgments forced. All judgments $\mathcal{J}$ are sampled uniformly at random, that is, $p(\mathcal{J}|\mathcal{A}, i) \sim p(\mathcal{J})$ whatever $\mathcal{A}$ stands for. Typically users $\mathcal{U}$ and items $\mathcal{I}$ are sampled first as randomly as possible over a larger set, and then judgments are sampled uniformly over $\mathcal{U} \times \mathcal{I}$. In some domains (such as music) the users can form an opinion on the fly for items they had not experienced before, whereas in others (e.g. books, movies, etc.) they can only judge items they were previously familiar with. The aforementioned CM100k dataset [4,5] is an example of the former kind.

$\mathcal{J}$ can be split uniformly at random into $\mathcal{J}_{train}$ and $\mathcal{J}_{test}$. The test sample is therefore uniformly distributed over $(\mathcal{U} \times \mathcal{I}) \setminus \mathcal{J}_{train}$ –to be more precise, it is distributed as a multinomial with uniform categorical probability for all items. Metric estimates are therefore unbiased in this setting. The shortcoming is however that by having a forcefully (and uniformly) obtained $\mathcal{J}_{train}$, the setting does not that faithfully represent realistic tasks where the system has to cope with an input $\mathcal{J}_{train}$ full of biases involved in free user activity.

However if judgments include information about the set of items that users were familiar with, it is possible to use this set as a reasonable proxy of natural free user feedback of the kind available to a real system [4], that is, as the training set $\mathcal{J}_{train}$. In order for the metric estimates to be unbiased, we should subsample $\mathcal{J}_{test}$ in such a way that $p(\mathcal{J}_{test}|\mathcal{R}, \neg\mathcal{J}_{train}, i)$ is constant on i. It is easy to see that the largest test set meeting this condition is obtained

by sampling test judgments among relevant non-training judgments with probability:

$$p(\mathcal{J}_{test}|\mathcal{J},\mathcal{R},\neg\mathcal{J}_{train},i) \leftarrow \frac{1-T}{1-T\,p(\mathcal{J})}\cdot\frac{1-p(\mathcal{J}_{train}|\mathcal{J},\mathcal{R},i)\,p(\mathcal{J})}{1-p(\mathcal{J}_{train}|\mathcal{J},\mathcal{R},i)}$$

where $T=\max_{j\in\mathcal{J}}p(\mathcal{J}_{train}|\mathcal{J},\mathcal{R},j)$ is the maximum training ratio over the relevant judgments of all items.

Non-relevant test judgments do not play any part whatsoever in the evaluation procedure (at least for all the range of metrics we are considering), so the test sampling probability of non-relevant judgments does not really matter, and we may as well sample none: $p(\mathcal{J}_{test}|\mathcal{J},\neg\mathcal{R},\neg\mathcal{J}_{train},i)\leftarrow 0$.

5.3 Online AB Testing

In online AB –or multiple variant– testing [24], the training data $\mathcal{J}_{train}$ is typically collected from spontaneous user activity in a live system, and the test data $\mathcal{J}_{test}$ is sampled from user interaction with recommendations output by the systems to be evaluated. In terms of representativeness, the setting cannot be more suitable, mostly if the test is performed on the final production platform. The obvious limitation of online evaluation is that it needs to be repeated altogether every time a new system is to be tested.

Depending how the test feedback is collected, we may consider two broad cases: free and forced sampling, which we discuss next.

5.3.1 Free test feedback. Users interact spontaneously with the recommended items, generating implicit evidence of their preferences, such as clicks, shares, downloads, purchases, etc., which can be interpreted as judgments. The distribution of test judgments over items is therefore subject to various biases, such as typically relevance and positional biases: users are commonly prone to interact with relevant items, and items placed at prominent places in the user interface, furthermore subject to complex inter-dependencies (e.g. redundancy) between items [9].

The setting does not meet any of the conditions stated in section 4 for unbiased metric estimation, that is, we definitely do not have $p(\mathcal{J}_{test}|\mathcal{R},\mathcal{T},i)\propto p(\mathcal{J}_{test}|\mathcal{R},\mathcal{T})$. However, to the extent that all systems are subject to the same biases, the qualitative system comparison can be expected to be fair. In particular, in the so-called interleaving approach [13,22] the items output by the tested systems are merged into a common ranking, in such ways that all systems are exposed to the same biases and inter-dependencies, whereby we should have $p(\mathcal{J}_{test}|\mathcal{R},R_{\theta_1})\sim p(\mathcal{J}_{test}|\mathcal{R},R_{\theta_2})$ for all systems $\theta_1,\theta_2\in\Theta$ participating in the evaluation, and we should therefore get fair qualitative system comparisons. Yet if user engagement in feedback was biased to any external condition such as, say, fashion, then a system biased towards recommending trendy items would be unfairly rewarded in the experiment.

5.3.2 Forced test feedback. Some user group $\mathcal{U}$ (e.g. raters, crowdsourced workers, etc.) is asked to judge all the delivered recommendations, typically some top k. In this case, since $\mathcal{J}_{test}\supset R$ for all the tested systems, we have $p(\mathcal{J}_{test}|\mathcal{R},R)=1$, whereby we get unbiased –and in fact quantitatively accurate– metric estimates: $\hat{P}\sim P$ and $\hat{Recall}\sim Recall$. This option would be therefore robust even to external biases. Of course if the collected feedback were later used for offline evaluation of systems that did not participate in the online experiment, the evaluation would unfairly favor algorithms that are similar to the ones the online experiment was originally carried out with (e.g. as a general trend, the evaluation would penalize innovative algorithms).

6 TARGET SAMPLING

Taking all unjudged user-item pairs as the target set $\mathcal{T}\leftarrow(\mathcal{U}\times\mathcal{I})\setminus\mathcal{J}_{train}$ in offline evaluation is the most common (explicit or implicit) option in the literature [11,23], to which we shall refer as *full targets*. Slight variations exclude from the target set users or items that lack data in the training and/or the test set, but these make negligible differences in practice [1], whereby we shall disregard them here. We analyze two common target sampling protocols that might make a substantial difference: a) taking only the test sample as the target set, and b) taking a fixed number of unjudged (or non-relevant) target items for each user, in addition to the test sample [1,8]. We shall refer to them as *test targets* and *user-flat targets* respectively. Both can be combined with any judgment sampling approach described in section 5.

6.1 Test Targets

The test-targets option $\mathcal{T}\leftarrow\mathcal{J}_{test}$ takes the smallest possible target set consistent with the assumptions stated in section 2. This option trivially ensures that the test sample is uniformly distributed over the target set, regardless of the judgment sampling approach: $p(\mathcal{J}_{test}|\mathcal{R},\mathcal{T},i)=p(\mathcal{J}_{test}|\mathcal{R},\mathcal{J}_{test},i)=1$. We hence remarkably find that we get fair comparisons in all metrics, regardless of the judgment sampling and split protocol.

The setting has however two major potential drawbacks. First, the representativeness can be seriously compromised: the target set is restricted to an extremely small and potentially biased item sample that is far from representing the wide range of items a real system needs to decide upon. This shortcoming is in fact shared by any configuration that takes a subset of the full potential target space, as a matter of degree according to the size of the subset.

And second, with a too small test target to choose from in building rankings, personalized algorithms are commonly unable to recommend the number of items required by the metric cutoff to all users. This raises the need to decide how to account for the coverage shortfall. Three main options are possible: a) counting the missing recommendations as non-relevant; b) filling the gap with random recommendations –or some other fallback option; and c) forgiving the shortfall by adjusting the output size expected by the metric to the size achieved by the system. Each of these options may be unfair in some way or other, whereby it is best in general to avoid the coverage loss altogether as far as possible.

6.2 User-Flat Targets

Flat targets [1,8] can be implemented by sampling a fixed amount of unjudged and/or non-relevant items for each user. If the sampling is uniform over items, it is not difficult to see that the procedure broadly preserves the item-wise biases present in the full-targets option, since item judgments in the target set are sampled from the same distribution as in the whole $\mathcal{J}$. Popularity biases, for instance, remain essentially unaltered, as was shown in [1]. Unless of course the amount of sampled targets approaches zero, in which case the configuration becomes test targets.

We have explored multiple other target sampling approaches in between test and full targets, but we find it impossible to produce (or consistently approximate) a uniform distribution $p(\mathcal{J}_{test}|\mathcal{R},\mathcal{T},i)$, mainly because relevance knowledge outside $\mathcal{J}$ would be needed, which is unavailable by definition in the experiment.

7 EXPERIMENT ON SYNTHETIC DATA

7.1 Experimental Approach

We test whether the conclusions of our theoretical analysis hold using synthetic data. In particular, we do a null hypothesis check: if user preferences were random, any possible recommendation algorithm should be just as effective as random recommendation –no more no less. Hence, if we randomly generate artificial user preferences, any experiment that returns thereupon a significant difference for some algorithm compared to a random recommendation is not fair: it is biased in favor of the algorithm that appears to be more effective than random.

We generate such synthetic data as follows. Assuming a user-item space $\mathcal{U} \times \mathcal{I}$ of a certain size, we randomly generate simulated preferences $\mathcal{R}$ from a Bernoulli distribution with a given constant prior relevance density $p(\mathcal{R}|u,i) \leftarrow p(\mathcal{R})$ for every $(u,i) \in \mathcal{U} \times \mathcal{I}$. Then, we generate judgments $\mathcal{J}$ with a given global judgment density $p(\mathcal{J})$ and biased distribution over users and items, namely a) a typical long-tail shaped judgment distribution $p(i|\mathcal{J})$ over items $i \in \mathcal{I}$, and b) a non-uniform conditional distribution $p(i|\mathcal{J},j)$ over different items $i.j \in \mathcal{I}$ –in other words, with some typical clustering bias over "similar" items who please common users [3].

We can reproduce both biases altogether very easily using the rating data in a common public dataset such as, for instance, MovieLens 1M [10], by simply randomly reassigning a new binary relevance status to each user-item pair with probability $p(\mathcal{R})$. This way, user tastes are random, uniformly and independently distributed over users and items, while judgments are not. On the other hand, to simulate the forced judgments approach described in section 5.2, we can take a certain number of MovieLens ratings (interpreted as a sample of items user are familiar with, as described in 5.2.2) uniformly at random with some probability $p(\mathcal{J}_{train}|\ rating,u,i) \leftarrow p(\mathcal{J})$ as training judgments, and sample the test judgments uniformly at random outside $\mathcal{J}_{train}$ with probability $p(\mathcal{J}_{test}|\neg\mathcal{J}_{train}, u,i) \leftarrow \big(p(\mathcal{J}) - p(\mathcal{J}_{train}|i)\big)/\big(1 - p(\mathcal{J}_{train}|i)\big)$, which can be seen to produce a global judgment density of $p(\mathcal{J})$, uniform over items and users. It would also be possible to recreate the approach described in 5.2.1, with equivalent results to the ones we describe next.

7.2 Results and Discussion

Fig. 2 shows the results for offline evaluation in terms of P@10, including the three judgment split options for free ratings described in section 5.1, and all judgments forced for the approach described in section 5.2. All configurations use full targets, and we show additionally the test targets sampling for the most biased option: the uniform random splits. The experiments compare three non-personalized recommendations: ranking by rating count (popularity), positive rating count (positive popularity), and positive rating ratio (average rating, with Dirichlet smoothing as in [4]); and two collaborative filtering algorithms: non-normalized

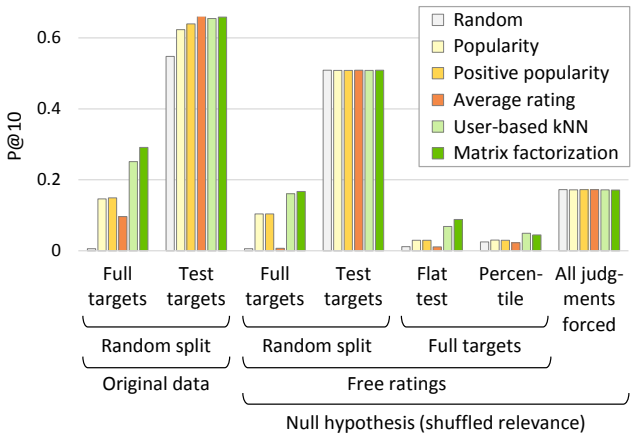

Figure 2: Simulated offline results based on MovieLens 1M data, taking $p(\mathcal{R}) \leftarrow 0.57$ (to get the same $p(\mathcal{R}|\mathcal{J})$ as in the original data) and $p(\mathcal{J}) \leftarrow 0.5$ as a simple setting. Random splits apply a 5-fold cross-validation, we take $t = 10$ test ratings per item in the flat test split, and $m = 10$ percentiles.

user-based kNN [3] (with $k = 50$ neighbors for full targets, and $k = \infty$ for test targets to avoid coverage losses) and implicit matrix factorization [14] (with 50 factors, $\alpha = 1$, $\lambda = 0.1$, 20 iterations). Algorithm settings are informally optimized for P@10 on MovieLens 1M (taking the rating values ≥ 4 as indicative of positive preference). We show only results for precision here; equivalent trends are observed for recall.

The results confirm the biases in the free ratings configuration: according to these experiments, popularity-based recommendation and collaborative filtering algorithms perform better than random recommendation, which does not make sense –i.e. the experimental outcome is unfair– considering that user tastes are random. The flat test and percentile options reduce unfairness, but the bias still remains to different degrees. We notice not only a bias towards popularity but also, and even stronger, towards the hypotheses (whatever they may be [3]) underlying the collaborative filtering algorithms. For instance, we may suspect the apparent effectiveness of kNN is due to some clustering structure in the judgments (which in the artificial null hypothesis is the same as in MovieLens ratings), while this structure is actually absent (we removed it) from user tastes by the null hypothesis. Interestingly, no bias in favor of or against the average rating is observed: it is deemed as effective as random recommendation. In contrast to free rating data, no noticeable bias is observed at all in the evaluation with forced judgments, thus confirming our theoretical analysis.

Also in agreement with our formal findings, we see that the test targets option removes all biases and is fair: all recommendations are at random level on the null hypothesis data. On the real MovieLens data, test targets would hint that popularity is overrated in usual experiments compared to the average rating, a finding that was already hinted in [4]; and collaborative filtering would also appear to be no better than trivial recommendation by the average rating. The weakness of test targets is however, as we discussed, in the potential representativeness shortcomings, whereby these observations on test ratings may need to be taken with a grain of salt.

As a final observation we may notice, somewhat disturbingly, that the unfair results with the random split (full targets) in the null hypothesis (shuffled relevance) have similarities to the corresponding results with the original MovieLens data. This might seem to suggest that a large fraction (as large as can be observed in the null hypothesis random split) of the measured effectiveness of popularity and collaborative filtering (and interestingly, not the average rating) in a typical offline experiment could be due to a pure bias in the evaluation.

8 CONCLUSIONS

Better understanding the guarantees and shortcomings of common evaluation procedures might uncover aspects in the effectiveness of recommendation algorithms that may have been overlooked, or even change our perception as to what the best algorithms really are. The main practical findings so far in our line of work reported here can be summarized in the following.

- Offline experiments with free user feedback are unavoidably unfair in some way or other. In general, the metrics are biased to favor the recommendation of popular items, regardless of whether popularity is truly effective or not [4]. Our experiments uncover other potential biases that favor assumptions on judgment structure (e.g. assuming pairwise user dependences as in kNN [3]), regardless of whether such structure is present or not in true user tastes.
- Test targets make any offline configuration fair, but compromise representativeness, and results in coverage shortfalls which may drive evaluation outcomes away from the real decision-making setting they aim to inform. From this point of view, our findings so far might advise against taking smaller than full target sets.
- Offline evaluation with randomly sampled judgments is fair, if the data is sampled and processed appropriately. Online AB testing should also grant fair comparisons, particularly if result interleaving [13,22] is used.

Work in progress at the time of this writing includes extensive experimentation with real data, complementing the experiments reported here. While some datasets with randomly sampled judgments are publicly available [4,20], building larger-size and/or higher-density datasets might be a worthy endeavor in this direction as well. We are also extending our analysis to evaluation perspectives and metrics that take into account novelty in addition to relevance [5].

ACKNOWLEDGMENTS

This work was supported by the Spanish Government (grant nr. TIN2016-80630-P).

REFERENCES

[1] A. Bellogín, P. Castells, and I. Cantador. 2017. Statistical Biases in Information Retrieval Metrics for Recommender Systems. *Information Retrieval* 20, 6 (Jul. 2017). Springer, Dordrecht, Netherlands, 606–634.

[2] T. Bertin-Mahieux, D. P.W. Ellis, B. Whitman, and P. Lamere. 2011. The Million Song Dataset. In *Proc. of the 12th International Society for Music Information Retrieval Conference* (ISMIR 2011). University of Miami, Miami, FL, USA, 591–596.

[3] R. Cañamares and P. Castells. 2017. A Probabilistic Reformulation of Memory-Based Collaborative Filtering – Implications on Popularity Biases. In *Proc. of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR 2017). ACM, New York, USA, 215–224.

[4] R. Cañamares and P. Castells. 2018. Should I Follow the Crowd? A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems. In *Proc. of the 41st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR 2018). ACM, New York, NY, USA, 415–424.

[5] R. Cañamares and P. Castells. 2018. From the PRP to the Low Prior Discovery Recall Principle for Recommender Systems. In *Proc. of the 41st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR 2018). ACM, New York, NY, USA, 1081–1084.

[6] P. Castells, N. J. Hurley, and S. Vargas. 2015. Novelty and Diversity in Recommender Systems. In: *Recommender Systems Handbook, 2nd edition*, F. Ricci, L. Rokach, and B. Shapira (Eds.). Springer, New York, NY, USA, 881–918.

[7] O. Celma. 2010. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer-Verlag, Berlin Heidelberg, Germany, 2010.

[8] P. Cremonesi, Y. Koren, and R. Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proc. of the 4th ACM Conference on Recommender Systems* (RecSys 2010). ACM, New York, NY, USA, 39–46.

[9] F. Guo, C. Liu, and Y. M. Wang. 2009. Efficient multiple-click models in web search. In *Proc. of the 2nd ACM International Conference on Web Search and Data Mining* (WSDM 2009). ACM, New York, NY, USA, 124–131.

[10] F. M. Harper and J. A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Trans. on Inf. Syst.* 5, 4 (Jan. 2016). ACM, New York, NY, USA.

[11] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. 2004. Evaluating Collaborative Filtering .Recommender Systems. *ACM Transactions on Information Systems* 22, 1 (Jan. 2004). ACM, New York, NY, USA, 5–53.

[12] J. M. Hernández-Lobato, N. Houlsby, Z. Ghahramani. 2014. Probabilistic Matrix Factorization with Non-random Missing Data. In *Proc. of the 31st International Conference on Machine Learning* (ICML 2014), 1512–1520.

[13] K. Hofmann, S. Whiteson, and M. de Rijke. 2013. Fidelity, Soundness, and Efficiency of Interleaved Comparison Methods. *ACM Transactions on Information Systems* 31, 4 (Nov. 2013). ACM, New York, NY, USA.

[14] 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 2008). IEEE Computer Society, Washington, DC, USA, 15–19.

[15] D. Jannach, L. Lerche, I. Kamehkhosh, and M. Jugovac. 2015. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction* 25, 5 (Dec. 2015). Kluwer Academic Publishers Hingham, MA, USA, 427–491.

[16] J. Konstan and G. Adomavicius. 2013. Toward identification and adoption of best practices in algorithmic recommender systems research. In *Proc. of the International ACM RecSys Workshop on Reproducibility and Replication in Recommender Systems Evaluation* (RepSys 2013). ACM, New York, NY, USA, 23-28.

[17] J. Koren. 2009. Collaborative filtering with temporal dynamics. In *Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD 2009). ACM, New York, NY, USA, 447–456.

[18] N. Lathia. 2010. *Evaluating Collaborative Filtering Over Time*. PhD thesis. University College London, London, UK.

[19] R. J. A. Little and D. B. Rubin. 1987. *Statistical analysis with missing data*. John Wiley & Sons, Hoboken, NJ, USA.

[20] B. M. Marlin, R. S. Zemel, S. T. Roweis, and M. Slaney. 2007. Collaborative Filtering and the Missing at Random Assumption. In *Proc. of the 23rd Conference on Uncertainty in Artificial Intelligence* (UAI 2007). AUAI Press, Arlington, VA, USA, 267–275.

[21] B. Marlin and R. Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proc. of the 3rd ACM Conference on Recommender Systems* (RecSys 2009). ACM, New York, NY, USA, 5–12.

[22] F. Radlinski and N. Craswell. 2013. Optimized interleaving for online retrieval evaluation. In *Proc. of the 6th ACM International Conference on Web Search and Data Mining* (WSDM 2013). ACM, New York, NY, USA, 245–254.

[23] G. Shani and A. Gunawardana, 2015. Evaluating Recommendation Systems. In: *Recommender Systems Handbook, 2nd edition*, F. Ricci, L. Rokach, and B. Shapira (Eds.). Springer, New York, NY, USA, Chapter 8, 265–308.

[24] A. Schuth, F. Sietsma, S. Whiteson, D. Lefortier, and M. de Rijke. 2010. Multileaved Comparisons for Fast Online Evaluation. In *Proc. of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (CIKM 2014). ACM, New York, NY, USA, 71–80.

[25] H. Steck. 2010. Training and testing of recommender systems on data missing not at random. In *Proc. of the 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (KDD 2010). ACM, New York, NY, USA, 713–722.

[26] H. Steck. 2011. Item popularity and recommendation accuracy. In *Proc. of the 5th ACM Conference on Recommender Systems* (RecSys 2011). ACM, New York, NY, USA, 125–132.