Cañamares, R., Castells, P. & Moffat, A. Offline evaluation options for recommender systems. *Information Retrieval Journal* (2020):23, 387–410

DOI: https://doi.org/10.1007/s10791-020-09371-3

Offline Evaluation Options for Recommender Systems

Rocío Cañamares · Pablo Castells · Alistair Moffat

Received: date / Accepted: date

Abstract We undertake a detailed examination of the steps that make up offline experiments for recommender system evaluation, including the manner in which the available ratings are filtered and split into training and test; the selection of a subset of the available users for the evaluation; the choice of strategy to handle the background effects that arise when the system is unable to provide scores for some items or users; the use of either full or condensed output lists for the purposes of scoring; scoring methods themselves, including alternative top-weighted mechanisms for condensed rankings; and the application of statistical testing on a weighted-by-user or weighted-by-volume basis as a mechanism for providing confidence in measured outcomes. We carry out experiments that illustrate the impact that each of these choice points can have on the usefulness of an end-to-end system evaluation, and provide examples of possible pitfalls. In particular, we show that varying the split between training and test data, or changing the evaluation metric, or how target items are selected, or how empty recommendations are dealt with, can give rise to comparisons that are vulnerable to misinterpretation, and may lead to different or even opposite outcomes, depending on the exact combination of settings used.

Keywords Recommender systems · evaluation · effectiveness metric · experimental design

Rocío Cañamares
Universidad Autónoma de Madrid, Spain

Pablo Castells
Universidad Autónoma de Madrid, Spain

Alistair Moffat
The University of Melbourne, Australia

1 Introduction

Evaluation is a cornerstone of research measurement in areas such as information retrieval (IR) and recommender systems (RS), and as a result, considerable effort has been put into developing techniques that provide robust and scrutable comparisons between alternative systems and alternative technologies (Gunawardana and Shani, 2015; Herlocker et al., 2004). The goal is typically to carry out a "system A versus system B" comparison, and identify whether A or B has the better performance. The two systems involved might be two quite different algorithms, or two variants of the same algorithm based on different implementations, or might even be two sets of tuning parameters being considered in conjunction with a single implementation.

Evaluation can be performed *offline*, using static data resources and corresponding evaluation metrics to compute numeric effectiveness measures that can be tuned for and/or compared; or can be carried out *online*, using a live system, and tracking user-related behaviors such as dwell-times, click-through rates, and purchase conversions. To assist with off-line evaluation, significant re-usable data resources have been developed via community investment and/or data release via sponsoring companies (Bertin-Mahieux et al., 2011; Harper and Konstan, 2016; He and McAuley, 2016). Businesses also develop their own in-house evaluation resources for offline evaluation. Here we undertake a detailed investigation into the methodologies used to carry out offline evaluation in the area of recommender systems. We consider a range of factors, summarizing the options that are available for each. In doing so, we provide explicit discussion of issues that have been either unconsidered or implicit in previous evaluations, and develop a taxonomy against which future evaluations can be categorized. Results from different investigations will then be able to be more accurately compared.

Our enumeration of decision points is supported by experiments that highlight our concerns. For example, we show that the split between training and test data, how target items are selected, and how empty recommendations are dealt with, can all substantially affect experiments, and hence argue that care needs to be taken so as to ensure that these risks are noted and commented on when researchers use such experiments to argue for their ideas.

2 Background

The Recommendation Task

A recommender system directly records or otherwise observes the interactions between a set of *users* and a set of *items*, as moderated by the intervening application (Adomavicius and Tuzhilin, 2005; Ricci et al., 2015). An interaction might consist of a user listening to a song, or giving a "star" rating for a movie, or clicking on a link, or reviewing a purchased product, or (in a negative sense) exiting a movie playback. The goal of the recommender system is to

build a predictive model of the tastes and preferences of each user, in order to be able to successfully recommend new (or replayed) songs, movies, and other products, and hence increase the engagement level of each individual user with the service.

The input of a recommender system is usually represented as a user-item matrix in which each non-empty cell corresponds to an observed interaction between the corresponding user and item. For simplicity, the different types of interaction are normally abstracted into a single (usually positive numeric) *rating* value, indicating the reaction that the user had as they viewed, consumed, or otherwise interacted with, that item. Low rating values are usually taken to mean "dislike", and/or "apathy", and/or "don't buy this"; and high ratings values taken to mean "like", and/or "would be happy to repeat the experience", and/or "would suggest this item to others".

Given such data, the recommender system seeks, for each of the users, to assign a *score* to every item. These scores – intended as a prediction of that user's likely rating reaction to that item – can then be decreasing-sorted to form a *ranking*, an ordered set of suggestions (Ricci et al., 2015). Out of the many thousands or millions of items included in the service, a small number of top-scoring recommendations, perhaps as few as three or five, are then offered to the user as future choices (Cremonesi et al., 2010). For example, an online book seller might recommend a list of future purchases to each user, based on their past purchase history and any reviews that user has lodged.

For maximum effect, the recommendations are usually *user-centric*, and tailored to the user's past history. It is also possible to construct a single *user-agnostic* (or *system-wide*) item ranking, based on global ratings data. The latter might be employed when insufficient user-oriented data is available – as a default starting point for newly-subscribed users, for example.

In IR terms, items can be regarded as being documents, and users as queries; the recommendation list then corresponds to a ranking of documents (Bellogín et al., 2013, 2017). In this framework, the ratings can be regarded as being relevance judgments, and used to score the ranked list. But there is also a critical distinction between these two activities: in many applications (such as book selling, for example) the system should avoid recommending the items that have already been rated, since either the user is unlikely to purchase the same item twice or, even if repeat purchase may be possible, a key purpose of the recommendation service is to help users discover items they would not find easily by themselves. A second distinction is that the ratings (judgments) originate with the community of users rather than being externally commissioned from either experts or crowd-workers following a process such as pooling over a collection of systems, the latter being the usual method for creating relevance judgments for an IR evaluation (Bailey et al., 2008; Harman, 2005; Kazai et al., 2013; Kutlu et al., 2018; Lu et al., 2016). These differences mean that effectiveness evaluation in recommender systems has taken a notably different path to evaluation in information retrieval systems (Bellogín et al., 2017; Gunawardana and Shani, 2015; Herlocker et al., 2004).

Recommender System Evaluation

In an *online* RS evaluation a live service is used to generate real-time recommendations for genuine users, and their reaction to those items is measured, perhaps via explicit pop-up surveys asking for ratings of the items offered, or perhaps by implicit signals such as selection clicks. In this methodology, different systems can then be evaluated in A/B-type comparisons, where the objective is, for example, to obtain the higher click-through or purchase rate (Hofmann et al., 2016).

In an *offline* evaluation the only information that can be used is a static snapshot of supplied ratings (Herlocker et al., 2004). In this type of experiment the available ratings are typically divided into *training* and *test* sets, with the decision as to whether a rating goes into training or test is an attribute of the (u, i) combination, not of u or i alone. The ratings in the training set are used as input to the recommender system so that it can build a model of user-item interactions. The system must then assign a score either to every item not in the training set associated with that user, or to every item in the test set for that user. There are a number of different ways in which the training/test split can be implemented (Bellogín et al., 2017; Cremonesi et al., 2010); these are canvassed in Section 3.

An offline evaluation also requires an *effectiveness metric*, a function that assigns a numeric score to a ranking, as an assessment of the quality, or accuracy, of a ranking compared to some suitable reference point (Valcarce et al., 2018). For example, one classic metric shared with IR is precision at depth d, the number of items in the first d recommendations for which positive ratings (values greater than or equal to some specified threshold) have been recorded in the test set. Other metrics and their relative merits are considered in Section 3. Note that the emphasis in an offline evaluation is usually accuracy in respect to the set of test ratings, but other facets, such as novelty or diversity, might also be taken into account (Castells et al., 2015).

The main drawback of offline evaluation is that the rating matrix is normally very sparse, with only a small fraction of the available (u, i) pairs known. That means it is necessary to decide what to do with recommended items that do not have ratings in the test set – whether they should be regarded as being "minimal" ratings and hence unhelpful suggestions to the user and of no utility, or whether they should be bypassed completely. This decision – considered in more detail in Section 3 – is not a simple one, because the observed ratings may not be an unbiased sample of the complete relevance matrix. That is, the rating matrix is likely to have a "missing not at random" distribution (Cañamares and Castells, 2018b; Marlin and Zemel, 2009; Steck, 2010).

In particular, if the rating data reflects past user actions, there is likely to be a strong *popularity bias* (Bellogín et al., 2017; Cañamares and Castells, 2018b; Steck, 2011): a relatively small fraction of the items accumulate the majority of the observed ratings (*short head*), while the rest have very few ratings (*long tail*). In this situation, missing cells are usually considered as non-relevant ratings, although, as already noted, it is also possible to restrict

Dataset	Users	Items	Ratings	Density
Movielens	6040	3706	1,000,209	4.47%
Netflix	480,189	17,770	100,480,507	1.18%

Table 1: Public datasets used in recommender systems evaluations.

recommendations to items with test ratings. The first option can benefit recommendations of popular items (Bellogín et al., 2017; Cañamares and Castells, 2018b; Cremonesi et al., 2010), because in a random split there is a correlation between the number of training and test ratings, and knowing which items have more test ratings reduces the probability of recommending an empty cell.

Typical Datasets

Several public datasets are available for RS evaluation. The MovieLens dataset includes ratings for movies on a 1 to 5 scale, by users of the MovieLens application, and was collected and published for the first time in 1998 (Harper and Konstan, 2016). The Netflix dataset contains data of a similar nature collected from Netflix subscribers, and was released to researchers in 2006 as part of the Netflix Prize contest.[1] Typically these datasets consist of one or more files of ratings, where each rating is either a tuple (u, i, r), with r a rating; or a tuple (u, i, r, t), where t is additionally a timestamp. Table 1 summarizes these three datasets.

Figure 1 shows the cumulative distributions of ratings per item and ratings per user for MovieLens and Netflix, and illustrates the long tail effect for both items and users. For example, around 50% of the users in MovieLens have each contributed 100 ratings or less, and about 45% of the items each have 100 ratings or less.

Ranking Items or Predicting Ratings?

An RS might also be regarded as generating a set of *rating predictions* rather than a ranking of items (Steck, 2013). Rating prediction evaluation is only appropriate if the recommender system's score function operates over the same numeric scale as the ratings themselves, in which case fidelity might be measured via the root mean square error (RMSE) between the set of predictions and the set of corresponding ratings. But not all recommendation methods can be evaluated via this protocol. For instance, user-agnostic recommending by averaging or taking the median of the ratings of each item could be evaluated by RMSE, but a scoring regime based on decreasing popularity could not.

Rating prediction-based RS evaluation was common in the past, but it is now broadly accepted that the ranking-based evaluation approach described earlier in this section is closer to the real recommendation task, because it

[1] http://www.netflixprize.com

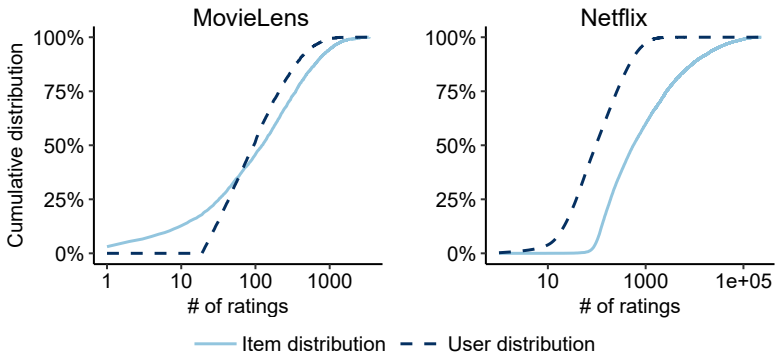

Fig. 1: Distribution of ratings per user and per item in MovieLens (left) and Netflix (right).

reflects the way in which items are offered to users without any attached score estimates (Cremonesi et al., 2010; Steck, 2013). In this work we focus solely on ranking-based RS evaluation.

Related Work

Evaluation was identified as a vital and non-trivial issue as the recommender systems field emerged towards maturity. Herlocker et al. (2004) provided an early comprehensive overview of evaluation methodology, covering the recommendation task definition variants, metrics, test data sampling, target item selection (implicitly considering condensed versus full rankings), online versus offline evaluation, and dimensions beyond accuracy. A decade and a half after its publication Herlocker et al.'s overview remains important; and many of the ideas and analyses it contains have been periodically rediscovered or reinvented. Gunawardana and Shani (2015) provide a useful summary covering similar issues, expanding on additional aspects such as robustness, privacy, and statistical power, providing a good starting point for initiation into RS evaluation practice.

These surveys discuss both rating prediction error and ranking metrics, with the shift from the former to the latter being argued for soon afterwards. Cremonesi et al. (2010) were particularly influential in this line, including proposing an intermediate option between ranking all items or only rated items (the "1 plus n" approach) in which a certain number of unrated items are added to the pool to be ranked by the evaluated systems. Shortly after, Bellogín et al. (2011) report illustrative experiments comparing results when adding all unrated items to the pool versus just a certain number versus none, along with some other options. Steck (2013) further analyzed the implications of this experimental setting and found an example where the comparison between two algorithms differed depending on whether all unrated items or none were added to the pool.

Bellogín et al. (2017) expanded their earlier work into a systematic description of ranking pool selection, and discussed in depth the issues arising in the adoption of IR metrics for RS evaluation, most notably the emergence of popularity biases. The adaptation of IR methodology to recommendation was reexamined by Valcarce et al. (2018), who both addressed the choice of metrics and metric depths according to the robustness to test rating sparsity, and also considered discriminative power. The concern in regard to popularity bias was specifically addressed by other authors, who sought to verify, measure and avoid such biases (Jannach et al., 2015; Marlin and Zemel, 2009; Steck, 2010, 2011), or to explain them (Cañamares and Castells, 2018b). Following up on some of these findings and drawing from related work in machine learning and statistics, a recent strand of research has addressed the bias in offline evaluation as an issue of mismatch between the data gathering policy (for example, free user interaction with a deployed system) and the item selection by the recommendation algorithms to be evaluated. Building on this perspective, techniques such as inverse propensity scoring have been explored to reduce the biases in evaluation (Gilotte et al., 2018; Gruson et al., 2019; Swaminathan et al., 2017; Yang et al., 2018) and in the algorithms being evaluated (Schnabel et al., 2016).

Divergence across experimental design and hence across experimental results has endured in the literature in spite these various cautions, and prompted further efforts towards clarification and shared consensus as to how state-of-the-art algorithms truly compare to each other. Said and Bellogín (2014), for example, undertook an extensive cross-comparison of public implementations of algorithms, datasets and evaluation options for ratings splitting and ranking pool selection. The main outcome of that study was to confirm the discrepancies between implementations of the – supposedly – same algorithms and metrics across different toolkits. Other recent activities make it clear that the community regards evaluation as an open issue in many aspects, with further effort needed (Ferro et al., 2018).

In a very recent paper, Dacrema et al. (2019) sought to reproduce the results reported in a suite of new RS system proposals, with rather disappointing results: some of the software systems were not available for re-testing; some results could not be reproduced (for example, because parameter settings or data splits had not been adequately disclosed); and some could be reproduced, but yielded results that were inferior to additional baseline systems not included by the proponents of that particular system. Only one of the implementations that were tested was competitive with well-tuned competitors.

Our work here continues this established thread of investigation and reflection, providing a global and systematic vision of a set of experiment configuration options that have either not been explicitly considered as settings of concern, or that deserve further analysis and attention, and serves as a complement to that of Dacrema et al. (2019). For instance, while user and item coverage have been considered as dimensions to watch out for, their potential to directly distort system comparisons has not been recorded previously in the literature. Metric aggregation functions other than arithmetic averages

are also rarely considered. Other aspects such as the split ratio, target item selection, and metric depth are dealt with in prior work but in generally different directions, and not as options than can change the comparative outcomes of evaluations in the sense in which we examine them here.

3 Methodological Decisions

As noted, several decisions must be taken when running an offline RS experiment. To provide a structure to those decisions, we now describe in detail the offline evaluation process and the choices that researchers are faced with at each stage of their experimental design. We divide the decisions according to four main phases: (A) data configuration; (B) recommendation; (C) measurement; and (D) system comparison; and note that there are several choices to be made in connection with each phase. In all our analysis, we assume recommendation is regarded as an item ranking task, the common understanding in the field through the last few years.

Decision A1: Data Selection

Experimenters must first choose a dataset, often a pragmatic decision based on availability. The use of public datasets is, of course, beneficial in terms of reproducibility. But sometimes the dataset is filtered in some way, altering it to suit the particular experiment. For example, users and/or items with fewer than some minimum number of ratings might be removed (Steck, 2013); and it is also not uncommon to remove some highly popular items (Cremonesi et al., 2010), so as to mitigate against popularity bias. In cases where this is done, both the thresholds used and the resultant dataset sizes (along the lines of Table 1) should be reported as part of the experimental description. It is also helpful if such filtering scripts can be made available, so that others with access to the same initial dataset can apply identical restrictions.

Decision A2: Train/Test Splitting

Offline evaluation methodologies then divide the selected data into Train and Test. The first set is used as input while model parameters are being tuned, the second is "held out" and used for measurement, after the model has been finalized. There are a many of ways to carry out the required division (Gunawardana and Shani, 2015):

- assigning ratings to Train with a specific random probability, for example 50:50 splitting, or in an 80:20 arrangement multiple times for five-fold cross-validation;
- assigning ℓ randomly-selected ratings per user to Test as a leave-ℓ-out split, assuming that every user has at least ℓ ratings available, see A1;

– assigning ℓ ratings per item to Train to mitigate popularity bias (Bellogín et al., 2017);
– assigning the oldest ratings to Train and the newest to Test as a temporal split, based on some nominal time cutoff applied to every rating to represent "now"; or
– combinations of these four options, for example, splitting the ratings of each user using a temporal split.

Decision B1: Selecting Users

After selecting and splitting the data, Train is used to build a recommendation model, and predictions are made for the user-item pairs it does not contain. Users with no (or too few) training ratings might be suppressed at this stage, since personalized recommenders would be unlikely to be able to assign scores. But those users could also be retained and a user-agnostic prediction mechanism employed to cover them – as might occur in a commercial service, for example.

Decision B2: NC Items

The recommendation outcome is a list of tuples (u, i, s), where s is a numeric score, and i ranges over all of the items not in u's Train set. Regardless of the ratings in Train, some algorithms may not be able to compute a score for all of the remaining items. For example, in the case of "k nearest neighbors" (kNN) approaches, if an item i has not been rated by any of u's neighbors, the pair (u, i) is not scored. We refer to these as non-computable (NC) items, and the researcher must decide how to handle them. One option is to remove them from the result set, even if they appear in Test; another is to assign a score computed via a different mechanism, for example a minimum score or a global user-agnostic score. Note that this is a different question to the one discussed in B1 (users not having items in Train), and that in the NC case any user-agnostic scores that arise must be converted to the same scoring scale as the organic results (somehow) so that they can be properly integrated into u's set of tuples (u, i, s).

An additional problem arises when NC items proliferate and the algorithm is unable to deliver recommendations for several users. This situation requires awareness and explicit decisions on the part of the experimenter in order for the experimental outcome to make proper sense. We discuss this further below, see Decision C4.

Decision C1: Condensed or Full

When measuring a set of RS rankings, a critical first question is whether to *condense* them by removing the items i for which (u, i) is not present in u's

Test set (Steck, 2013). Doing so means that there are no unrated items present when computing the value of the chosen metric (see Decision C2, below), and perhaps gives confidence in that regard, but also means that many – perhaps the great majority – of the (u, i, s) tuples generated are discarded. Significant distortion to the metric score might be introduced as a result, since the task in the experiment no longer represents the task the algorithm serves in production setting – in the latter situation all items are scored and considered for presentation. Condensed rankings are also prone to result in coverage shortfalls, as we discuss as Decision C4, below. Condensed rankings may nonetheless have properties that make them useful as an experimental option, chief being that uncertainty about item relevance is avoided. Condensed rankings have also been found to have good properties in avoiding experimental bias (Cañamares and Castells, 2018a).

If the rankings are not condensed and are used in full, a followup decision is then required, and that is how to treat the (u, i, s) pairs in the ranking that do not have a Test rating. One option is to assign a minimal (that is, non-relevant) rating to them, and count them as "zeros" in terms of the metric computation (Cañamares and Castells, 2017; Steck, 2013). But this is arbitrary, and the same items could equally well be assigned "full-relevance" labels. A third option is to employ a metric in which the extent of the measurement uncertainty can be computed, and report that quantity as part of the evaluation.

Decision C2: Choice of Metric – Full Rankings

The usual approach to evaluating full rankings is to apply an IR effectiveness metric, and compute a numeric score using the Test ratings as *relevance judgments* (Herlocker et al., 2004). In IR evaluations judgments are typically either *binary* or *graded*. Binary judgments of "not-relevant" (zero) or "relevant" (one) are used for metrics such as precision and average precision (AP). When categorical Test ratings (such as "stars") are being used as relevance judgments with these metrics they must be binarized in some manner, another key decision point. For example, ratings $1 \le r \le 5$ might be binarized via the criteria $r \ge 4$, with the threshold value of 4 another experimental decision that might affect the outcome.

Ratings might also be used as ordinal relevance categories in a graded relevance situation, such as when NDCG (Järvelin and Kekäläinen, 2002) or rank-biased precision (RBP) (Moffat and Zobel, 2008) are being computed. If so, a *gain mapping* is required, converting each ordinal category to a numeric utility. One commonly-used function maps grade $1 \le r \le r_{\max}$ to a numeric gain $g(r)$ via the powers of two, $g(r) = (2^{r-1} - 1)/2^{r_{\max}-1}$ (Chapelle et al., 2009); scaling into the full range $0 \ldots 1$ is also possible via $g(r) = (2^{r-1} - 1)/(2^{r_{\max}-1} - 1)$. Other mappings are also possible, with binarization always available as a seemingly straightforward option – except, that is, for the need to choose a threshold, as noted in the previous paragraph.

Top-weighted metrics are usually preferred, so that incorrect estimates early in the ranking (high estimated scores, but low Test ratings) are penalized more heavily than errors later in the ranking. It is also usual for evaluation to be truncated at a relatively shallow depth d, rather than computed over the full length of the ranking (Cremonesi et al., 2010), to match the typical use-case of a recommender system, in which a user is presented with a set of d highly-scored items as suggestions, and is likely to keep their exploration within that set. It is also desirable for the metric score to reflect the experience of the user as they peruse the recommendations. Moffat et al. (2017) explore the implications of user behavior on IR metric design.

In RS evaluations, common metrics include Prec@d (which is not top-weighted) and NDCG@d (Järvelin and Kekäläinen, 2002) (which is top-weighted, but requires knowledge of the d highest ratings associated with items not in Train, so that an "ideal ranking" can be formed), for values of d such as $d = 5$ or $d = 10$. Note that NDCG cannot be computed unless user u has at least one rating r in Test for which the gain value $g(r)$ is greater than zero. This restriction is sometimes bypassed by defining NDCG to be zero for those users, and sometimes bypassed by removing those users from the experimental evaluation. The two options have different effects on the aggregate metric score.

The clear challenge with evaluation via full rankings is data sparsity (the final column of Table 1). If all non-Train items are scored and included in the metric evaluation for user u, then it is entirely possible that many, or even all, of the top-d recommendations will not have corresponding ratings. Assigning these as "not relevant", or "grade 1" (and gain zero) might greatly underestimate the numeric value of the metric. Indeed, if two systems are being compared in this way, and the stronger of the two does indeed populate the head of the ranking with better recommendations, it might in fact be disproportionately disadvantaged.

Decision C2: Choice of Metric – Condensed Rankings

Condensed rankings have also been used in IR evaluations, with mixed success (Sakai and Kando, 2008). In IR, unjudged items are usually assumed to be in the minority near the top of any ranked list, because of the pooling process that is used to generate the judgments; and to primarily occur at lower positions in the ranked list, where their weight in the metric score is lower. But in RS evaluation, the condensing process may remove high-scoring items as well as low-scoring items. Indeed, computing an IR metric at depth d over an RS condensed ranking might be quite misleading, since the bias towards positive ratings may mean that all of the run scores that get computed end up being very close to one, thereby presenting a rose-tinted view of the user's actual experience (Moffat et al., 2017).

Rather than computing an IR metric for condensed rankings, another option is to ask how dissimilar the ranking of Test items induced by the scoring function is to the ordering imposed on those same items by the ratings, and

compute a *correlation coefficient*. If RS scores are highly correlated with the Test ratings, then it can then be inferred that the system is providing an accurate estimation. Traditional correlation approaches such as Kendall's τ have the disadvantage of not being top-weighted, and of penalizing all discordant pairs equally. Top-weighted coefficients can also be computed. For example, NDCG (Järvelin and Kekäläinen, 2002) compares a given ordered ranking with an "ideal" ranking of the same items; and hence, when applied to a condensed list, can be thought of as computing a correlation coefficient. Other IR metrics have also been adapted to provide top-weighted coefficients, including average precision (AP) (Yilmaz et al., 2008).

Webber et al. (2010) describe *rank-biased overlap* (RBO), which computes the expected overlap between two sequences "A" and "B" assuming that the user always looks at the first pair of items, one element from each of A and B, and proceeds from the d th pair of items to the $d+1$ st pair with conditional probability ϕ, where ϕ controls the extent of the top-weightedness. When ϕ is small, for example, $\phi = 0.5$, the expected depth reached by users is $1/(1-\phi) = 2$, and discords near the head of the lists give rise to a high penalty. On the other hand, when ϕ is large the expected depth is higher, and if the two lists have the same first few items, even if in a different order, RBO will remain relatively high. To employ RBO, sequence "A" is the set of items in Test, decreasing-sorted by the recommender's score; and sequence B is the same set of items, but sorted by decreasing rating as primary key, and decreasing RS score as secondary key, so that ties on rating are handled in a deterministic manner. In this regard RBO compares to an "ideal" ordering in the same way as does NDCG.

Decision C3: Metric Value Aggregation

Once a score has been computed for each user, there is still the question of representing performance over a set of users. One obvious possibility is to compute an unweighted arithmetic mean, in which all users are treated as being equally important. Other central tendencies can also be employed, including the median and the geometric mean. Alternatively, each user's score might be weighted according to the number of Test elements associated with that user (Steck, 2013), or some other volume-related factor, so that users with many ratings in Train and Test have more influence on the overall system score than those with few ratings. A commercial provider might wish, for example, to have high-volume users exerting more weight than do low-volume users.

Risk aware aggregation mechanisms might also be employed (Dinçer et al., 2014a,b). These approaches provide different weightings to "wins" than they do to "losses", and are intended to ensure that a System A versus System B transition does not bring with it the possibly high reputational damage that might arise if even a small minority of the users receive substantially inferior performance as a result of the change.

Score standardization relative to a larger set of reference systems is another transformation that might be considered as part of the processing pipeline used to compare scores when carrying out an A versus B batch evaluation (Webber et al., 2008).

Decision C4: Coverage Shortfall

Certain recommendation algorithms may fail to deliver any recommendation at all for some users, typically as a result of lack of sufficient data. For example, user-based kNN may fail to find any neighbor of u who has rated any recommendable target item. That is, there may be users for whom all target items are NC, as described in B2, above. The situation may be exacerbated if a strong restriction on the designation of target items is applied in offline evaluation – as is the case when condensed rankings are employed (see Decision C1 above) (Cañamares and Castells, 2018a).

In such cases the experimenter has two options: they can "forgive" the lack of coverage, by averaging the metrics only over the users who did get recommendations; or they can average over all users by declaring the uncovered users to have a metric score of zero. While the full average seems a reasonable default option, the reduced "forgiven" average might also be appropriate in some circumstances. An additional alternative, circumventing the problem, is to handle all NC items via a secondary scoring regime (B2, above), even if that means the whole ranking for some users is generated that way.

Decisions in regard to user coverage shortage and the difference between the two averaging options might be regarded by the experimenter as being relatively inconsequential. But they can greatly distort the results and potentially lead to incorrect experimental conclusions. In particular, forgiveness likely boosts the apparent effectiveness of a short-coverage algorithm, since it is permitted to refuse to deal with "difficult" users, and obtains an advantage over algorithms that produce recommendations for all users.

Assuming metric m is aggregated by arithmetic mean over users, it is easy to switch between full and reduced averages by $m_{\text{reduced}} = \text{User-coverage} \cdot m_{\text{full}}$, where m_{full} and m_{reduced} are the metric values obtained by a full and reduced average respectively, and User-coverage is the fraction of users covered. But both approaches have disadvantages: the full average penalizes empty recommendations as much as it penalizes totally nonsensical recommendations, even though a void output is not delivered to anyone, and does no actual harm; a reduced average places no penalty on the failure to bring service to some users. As a result of these concerns, we recommend that researchers check for coverage as a complementary dimension to recommendation quality, and habitually report coverage alongside (and separate from) effectiveness scores.

The lack of coverage might be quite insidious, since the metric scores are typically computed to some depth d, and there might be fewer than d non-NC items found for user u. Moreover, most metric computation software deals with any missing part of the ranking by assuming a necessary quorum of

non-relevant items. Truncated rankings may thus result in a degradation of the perceived effectiveness of the algorithm, an effect which might similarly be unnoticed by the experimenter. It may thus be appropriate to employ a finer and more informative coverage metric than the User-coverage ratio above, defined as:

$$\mathsf{Coverage@}d = \frac{1}{d|\mathcal{U}|} \sum_{u \in \mathcal{U}} \min(d, |R_u|)\,,$$

where $\mathcal{U}$ is the set of all users, R_u is the recommendation delivered to user u, and d is the metric depth for the quality measures under examination. It is easy to see that the two coverage metrics are related by User-coverage = Coverage@1. Consideration of this Coverage@d score will highlight measurement quality losses resulting from coverage problems. The *residual* value associated with weighted-precision metrics serves a similar role in IR (Lu et al., 2016; Moffat and Zobel, 2008; Moffat et al., 2017).

Decision D. System Comparison and Statistical Tests

As well as comparing systems according to their mean or median performance it is common to carry out a statistical test (Gunawardana and Shani, 2015; Sakai, 2016, 2018). If the reporting is of how many users each approach is best for, then the sign test should be used. Or, if the quantities being compared are the two arithmetic means, then the t-test is suitable. Subsampling can also be used to help establish confidence. For example, the pool of users can be sampled to create random subsets in a mode akin to bootstrapping, and a statistical test applied in each subset, with the final reported outcome the fraction of subsets for which the statistical test rejected the null hypothesis relative to some threshold α such as 0.05 or 0.01. Researchers and reviewers alike should agree that *not* carrying out a suitable statistical test is *not* a permitted option; and that reporting of the computed p values is preferable to binarization of them relative to a threshold such as $\alpha = 0.05$ (Wasserstein et al., 2019).

Implications for Reproducibility

As an immediate consequence of the enumeration of factors we have provided in this section, we suggest that researchers carrying our experimental evaluations of recommender systems take care to fully describe their experimental procedures, so that others might hope to replicate their results given access to the same data.

4 Experiments

We now explore the effect of the methodological decisions described in Section 3 and show that they can play a decisive role in system-versus-system

comparisons. To accomplish this, we employ two well-known recommendation methods: a user-based version of the k-nearest-neighbor algorithm (kNN) (Ning et al., 2015), and matrix factorization (MF) (Hu et al., 2008), making use of public implementations provided by the Ranksys library (see `http://ranksys.org`). In MovieLens, $k = 100$ neighbors and cosine similarity are used for kNN; and for matrix factorization the configuration parameters proposed by Cañamares and Castells (2017) are employed, namely 50 factors, $\lambda = 1$, $\alpha = 0.1$ and 20 iterations. The same settings are used in Netflix, except $k = 1000$ for kNN. Note that choosing the best parameter configuration for either approach is not an essential point here, since our objective is not to identify the "best" RS system, but to instead demonstrate in practical terms the effect that the evaluation design can have on an experimental outcome, regardless of what the two candidate algorithms might be.

Starting Configuration

The following settings, presented in the categories described in Section 3, are used as a starting point, and represent a completely typical RS experimental arrangement. The majority of these settings are then explored in the experiments that follow:

- A1 *Data selection*: the MovieLens and Netflix datasets are used.
- A2 *Train/test split*: the available ratings are separated into Train and Test using a random split with a split ratio of 0.5.
- B1 *Selecting users*: users are removed if they do not have any ratings in Train after the splitting process.
- B2 NC *items*: items without a score are not included in the ranking.
- C1 *Condensed or full*: the experiments commence with full-list evaluation, the option most commonly used in the literature.
- C2 *Choice of metric, full rankings*: Prec@d and NDCG@d are used at $d = 10$, with a binarized gain mapping $g_1(r) = 0$ if $r < 4$, and $g_1(r) = 1$ if $r \geq 4$, where $1 \leq r \leq 5$ for both datasets. If an item has not been rated its gain is 0. Rankings containing no ratings, or rankings in which the only ratings have gains of zero, are defined to have scores of zero. When Prec and NDCG display similar behavior, we only show NDCG.
- C3 *Score aggregation*: the unweighted arithmetic mean over users is used to compute aggregate Prec@d and NDCG@d scores.
- C4 *Coverage shortfall*: by default full metric averages over users are computed, including users to whom the system fails to deliver a recommendation and hence have zero scores.
- D *Statistical tests*: In all our experiments a paired two-tailed t-test is used to assess the statistical significance of metric comparisons between system pairs – the resulting p values are reported whenever they are material for the point made in our analysis and/or the corresponding findings. In these tests, we follow the same approach as is common in the evaluation of search results (Carterette, 2012), where in our case the users play the

Metric	MovieLens			Netflix		
	kNN	MF	p	kNN	MF	p
Prec@10	0.416	0.423	0.039	0.382	0.409	0
NDCG@10	0.452	0.455	0.897	0.416	0.442	0

Table 2: Performance of kNN and MF using the initial settings. For each dataset and metric, the better value of each comparison pair is shown in blue, and the third column in the group lists p values for a paired two-tailed t-test.

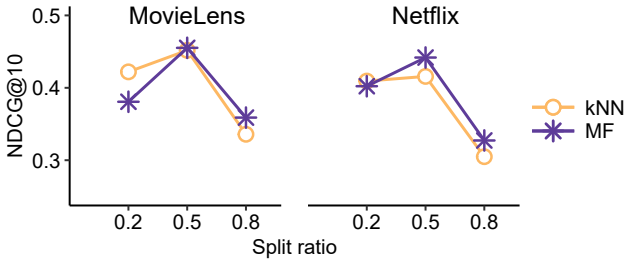

Fig. 2: System scores: NDCG@10 for kNN and MF varying (only) the test/train split ratio (A2). All other settings are as specified for the starting point. Similar behavior arises with Prec@10. The p values for a paired two-tailed t-test are essentially 0 for all the comparisons, except in MovieLens at a 0.5 split ratio where the p value is 0.897.

role of queries: the statistical tests measure the significance of the mean difference of a metric between two systems based on the series of metric value differences of the systems over individual users. To simplify the analysis, each metric value data point is computed over a single data split into training and test.

Table 2 shows the performance of kNN and MF when a system-versus-system evaluation is carried out using these typical experimental settings. Matrix factorization provides slightly better performance than kNN when measured using both metrics, and hence might be deemed to be the "winner" in this experiment. The outcome is consistent across the two datasets, with a slightly higher margin for Netflix. We now systematically explore the effect on the same experiment if each of the various experimental decisions is varied.

Decision A2: Train/Test Splitting

The split ratio of 0.5 yields Train and Test sets of approximately equal size. Other values are also sometimes used – in particular, 0.8 is common for a five-fold cross-validation experiment, and similarly 0.9 for a ten-fold experiment. Figure 2 shows how the NDCG@10 scores of kNN and MF change for lower and higher split ratios than the starting value 0.5. At a small split value kNN outperforms MF on both datasets, but when the split ratio goes over 0.5 the ordering is reversed.

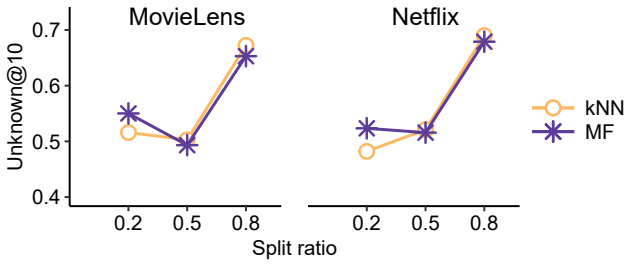

Fig. 3: Fraction of unrated items in the top ten positions, averaged across users, varying (only) the split ratio. The p values for a paired two-tailed t-test are essentially 0 for all the comparisons, except in MovieLens at a 0.5 split ratio where the p value is 0.006.

The actual value of NDCG@10 also changes (within each algorithm) as the split ratio is varied – it turns out that the maximum scores tend to be recorded at around 0.5. This behavior is the result of two factors acting in tension. On the one hand, the more Train ratings there are for a user, the more likely it is that the system will assess that user's tastes correctly, and make useful suggestions. On the other hand, since full-list evaluation is being employed, more Train ratings implies fewer rated items in Test, and only the latter positively contribute to the metric score.

Figure 3 shows the fraction of unrated items in the top-10 rankings for kNN and MF, and illustrates the problems caused by low rating density. There is no split point at which the fraction of unrated items in the rankings is less than 50%, and for both small and large split points, a great majority of the items being used in a full-ranking evaluation have unknown ratings – yet are counted as having a gain of zero when depth $d = 10$ non-condensed metrics are computed. That is, fixed-depth evaluation using full rankings means that a majority of the ranked items are assigned artificial gain values (of zero), risking the integrity of the evaluation.

Decision C1: Condensed or Full

The alternative is to condense the rankings, and work only with rated Test items. Figure 4 shows the scores that arise. The pattern of Figure 2 is gone, and now there is a steady improvement in scores as more ratings are made available for training. Moreover, in Netflix the qualitative comparison between kNN and MF is no longer sensitive to the split ratio. However, the cross-over between the two methods remains for MovieLens: kNN gives higher scores than MF when there are more Test ratings than Train ratings.

To gain a further insight into condensed lists, Figure 5 shows the position in the full ranking of each of the top ten items that make up the two condensed rankings, with each position's value an average across all users. In MovieLens, only the top three items in the condensed list are, on average, present in the

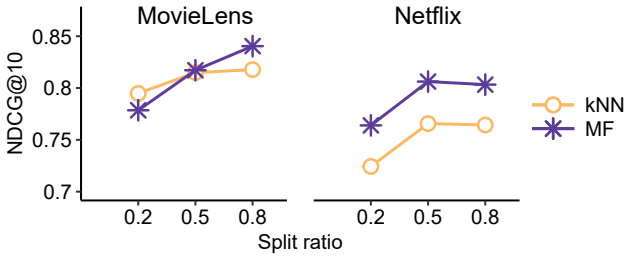

Fig. 4: System scores: NDCG@10 for kNN and MF as a function of the test/train split (A2), now using condensed lists (C1). The p values for a paired two-tailed t-test are essentially 0 for all the comparisons, except in MovieLens at a 0.5 split ratio where the p value is 0.041. Somewhat similar behavior arises with Prec@10.

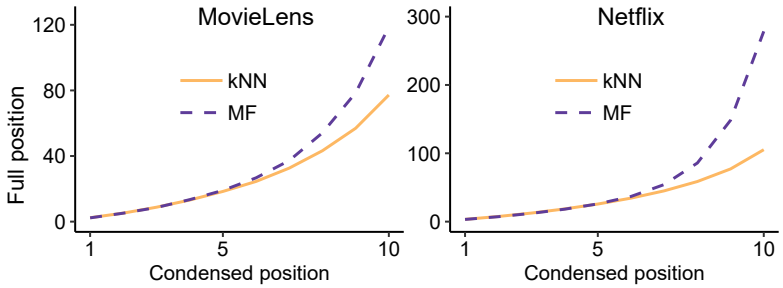

Fig. 5: Average position in the full ranking of each of the first ten items in the condensed ranking, using the "starting point" configuration. This experiment only includes users for whom ten or more Test ratings were available using both mechanisms (B1), so as to allow computation of all ten points over the same set of users.

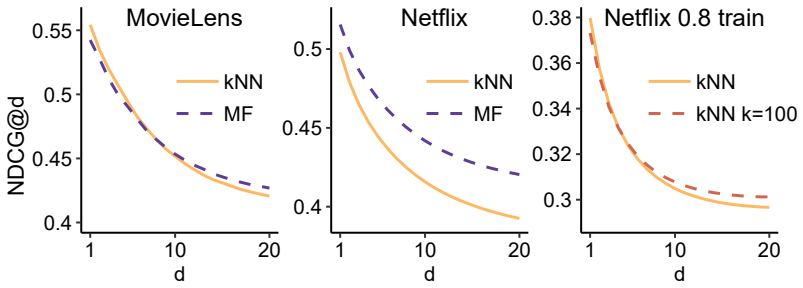

Fig. 6: System scores: NDCG@d for kNN and MF as d is varied (C2). All other settings are as specified for the starting point. The vertical axes are truncated.

top ten full list; and by depth ten in the condensed list, more than fifty items in the full list have been covered. This then raises a potential concern against the use of condensed lists: the evaluation considers a very different set of items to those that would be shown to the user in a real environment.

Metric	Gain mapping	kNN	MF	p
Prec@10	$g_1(r)$	0.416	0.423	0.039
NDCG@10	$g_1(r)$	0.452	0.455	0.897
NDCG@10	$g_2(r)$	0.393	0.389	0.002
RBP, $\phi = 0.8$	$g_1(r)$	0.410	0.410	0.390
RBP, $\phi = 0.8$	$g_2(r)$	0.339	0.332	0.011

Table 3: System comparisons: scores for kNN and MF in MovieLens when different gain mappings are employed with different effectiveness metrics (C2) on top-10 lists, and corresponding p values for paired two-tailed t tests. (The first two rows also appear in Table 2.) All other settings are as specified for the starting point. The better value in each pair is shown in blue.

One surprising aspect of Figure 5 is that the depth required to form the condensed MF list grows much faster than the depth required for kNN. That is, MF appears to be more willing to provide scores for novel items than is kNN, which seems to be more closely aligned with the set of items in Test. This willingness might mean that MF scores more poorly than it should on full rankings.

Decision C2: Choice of Metric

The metric should be chosen to reflect the likely user interaction with the generated ranking (Moffat et al., 2017). Then, once the metric has been selected, there is usually a parameter that must be set. In the case of Prec and NDCG, that parameter is d, the evaluation depth. Figure 6 shows the value of NDCG@d over different evaluation depths for kNN and MF. In MovieLens (left), when d is small, kNN provides superior performance, but as d increases, MF overtakes it. In Netflix (center), the comparison is not sensitive to the metric depth. However we can easily find a similar crossover in this dataset: a different configuration of kNN, just taking $k = 100$ neighbors instead of 1000, yields a different outcome to the base kNN configuration at a 0.8 split ratio (Figure 6, right).

The choice of gain mapping also has a notable effect. Table 3 shows average metric scores in MovieLens with the standard binarized gain mapping $g_1(r) = \mathbb{1}_{[4,\infty)}(r)$ defined as the "starting point", and compares those values with the multi-valued mapping $g_2(r) = (2^{r-1} - 1)/(2^{r_{\max}-1} - 1)$ that is often used in IR applications. The relative superiority between kNN and MF is again reversed by a single change in the experimental methodology. The differences g_1 and g_2 indicate that kNN does a better job than MF at ranking high ratings above moderate ones (as captured by g_2), while MF is better (with weaker significance) at ranking high and moderate ratings above low and missing ratings (as represented in g_1). This effect is confirmed by Kendall's τ and RBO scores – not included in our presentation here – which confirm that the condensed kNN rankings are more similar to the Test rankings than are the MF rankings.

Aggregation	Prec@10			NDCG@10		
	kNN	MF	p	kNN	MF	p
AM, unweighted	0.416	0.423	0.039	0.452	0.455	0.897
AM, Test weighted	0.591	0.586	0.123	0.619	0.611	0.001
AM, Test relevant	0.597	0.590	0.013	0.625	0.612	1.826×10^{-4}
GM, unweighted	0.294	0.308	–	0.320	0.332	–

Table 4: System comparisons using MovieLens, with kNN and MF scores computed using different aggregation techniques (C3), where AM indicates the arithmetic mean, and GM indicates an ϵ-adjusted geometric mean with $\epsilon = 0.01$. All other settings are as specified for the starting point. For each metric and aggregation technique, the better value of each comparison pair is shown in blue, and the third column in the group lists p values for a paired two-tailed t-test.

Decision C3: Score Aggregation

Table 4 shows the value of Prec@10 and NDCG@10 when different averaging methods are employed in MovieLens. Weighting the scores by the number of ratings associated with each user or weighting them by the number of positive ratings (≥ 4) associated with each user yield different numeric scores, and, more importantly, also reverse the system comparison. In the MovieLens data set there is a strong correlation between the number of ratings a user has generated, and the number of positive ratings they have lodged (a Pearson's ρ of 0.962), and it appears from this experiment that kNN benefits from having more ratings available. The lift in absolute scores is a result of two factors: first, more Test ratings implies more rated candidates in the full list, and hence fewer "presumed zero" ratings; and second, more Train ratings for those same users implies better quality predictions, since the algorithm has more information to work with. Note that a commercial service might prefer a weighted comparison, given that users with more historical ratings are also more likely to return to the service as future customers.

Table 4 also includes the geometric mean. Taking the arithmetic mean across users means that small differences between the systems for large scores have the same importance in the comparison as small differences when the scores are small. But the latter are relatively more worthwhile, an effect that has also been noted in IR evaluation Robertson (2006). One issue that arises with the geometric mean is that if any of the values in the set being aggregated is zero, so too is the geometric mean. To avoid this problem, we employ the *ϵ-adjusted geometric mean*, computed as $\exp(\sum_{i=1}^{n} \ln(x_i + \epsilon)/n) - \epsilon$. Use of the t-test is not appropriate in this case, because the statistic being reported is not the mean; that is why no p values were computed. But if we regard the critical part of the GM computation as being the transformation into log-space, then the t-test can be applied to the two sets of $\ln(x_i + \epsilon)$ values, and if this is done the two p values are 1.769×10^{-6} and 1.653×10^{-4} respectively.

Table 4 provides a clear warning – the four score aggregation techniques give notably different outcomes; worse, each of the four options might be argued for as being the preferred approach.

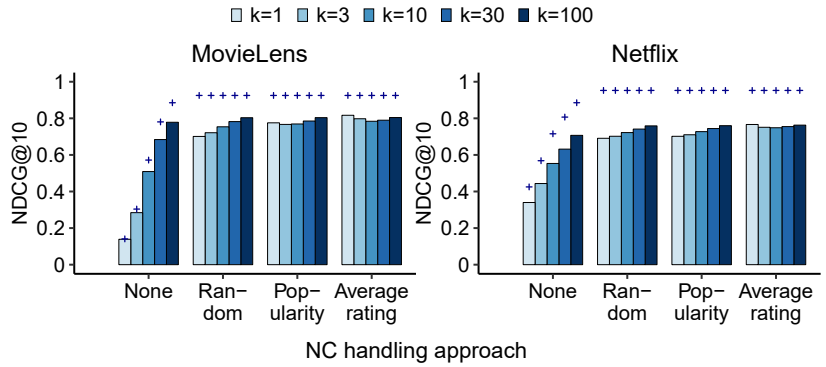

Fig. 7: Scores for NDCG@10 for kNN with $k = 1, 3, 10, 30$ and 100, with different approaches to NC items. Recommendation lists are condensed, and half of the ratings are now used in MovieLens. Blue crosses represent Coverage@10.

Decisions B1, B2: Selecting Users and Handling NC Items

All of the experiments reported so far excluded users with no Train ratings. When working with the MovieLens dataset, in which all users have more than 20 ratings, and a split ratio of 0.5, this restriction has little influence on the experimental outcomes. Only for small split ratios was a lack of Train ratings an issue, and in practice both kNN and MF can compute internal scores for almost all items. Hence, to observe the effect of the B1 and B2 decisions, we now make a more complex change to the starting configuration:

- A1 Half of the ratings in the MovieLens dataset are randomly selected, and then split between Train and Test, to increase the number of NC items. In Netflix many users have very few ratings already, so this operation is not necessary.
- B1 All users are included, even those with no Train data.
- B2 NC items. We compare four approaches: removing them from the recommendation list (*None*), as was done in the previous experiments; and adding them at the end of the non-null ratings in random order (*Random*); in decreasing order of Train ratings (*Popularity*); or in decreasing order of average rating in Train (*Average rating*).
- C1 To allow the NC items to exert influence on the score, the condensed list approach is used.

Five different versions of kNN are then compared, using $k = 1$, 3, 10, 30, and 100 neighbors, with the increasing number of neighbors decreasing the number of NC items. While not all the resultant configurations can still be regarded as being "typical", they nevertheless serve to illustrate the effect of NC items. Figure 7 shows the value of NDCG@10 (Prec@10 behaves similarly) when the NC items are handled using the four options.

If the NC items are removed from the recommendation list (*None* approach), considering more neighbors results in better performance. This is

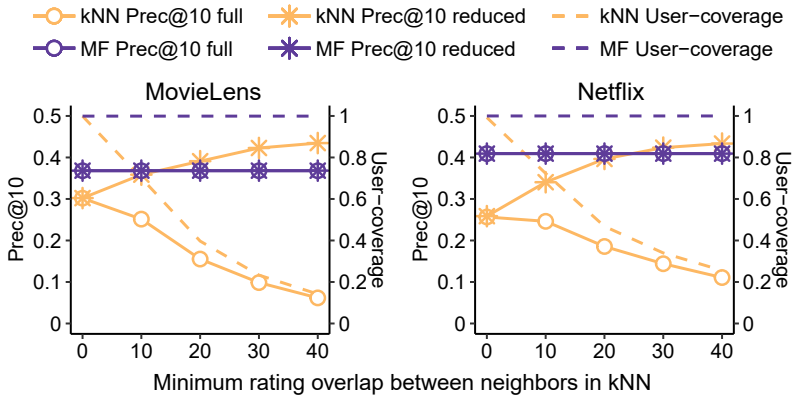

Fig. 8: System scores for Prec@10 (left axis) and User-coverage (right axis) as a function of the kNN requirement for a minimum rating overlap between neighbors. Precision is shown in two variants: full, and reduced average. The performance of MF is independent of the kNN parameter, and shows as horizontal lines. All other settings are as specified for the starting point.

because the overabundance of NC items results in coverage loss, shown in the figure by blue crosses representing Coverage@10 (see C4 in Section 2). This in turn hurts NDCG@10, as is apparent in the first block of bars. The same shift occurs with the *Random* approach, but to a lesser extent. The difference exposes the influence of NC items, and confirms that the smaller the number of neighbors, the greater the number of NC items. But the situation is not as clear-cut with the other two approaches. Now the change is not monotonic, and in the *Average rating* approach, the best performance is obtained when $k = 1$, meaning that using a non-personalized algorithm for many of the items is better than using kNN alone. This data provides another clear warning: NC items alone might determine the comparative performance of two algorithms.

Decision C4: Coverage Shortfall

Figure 8 shows an example where contradictory conclusions would arise in the comparison of the kNN algorithm ($k = 10$) against matrix factorization because of coverage. The kNN variant employs binarized ratings, with cosine similarity values accepted as valid only when that user has a minimum number of overlapping rated items. This is a common adjustment in kNN to avoid unreliable similarity measurements based on too few ratings, but has the side effect of users with too few neighbors failing to receive recommendations. Compounding the issue is that the excluded users are the "difficult" ones with few Test ratings, that is, for whom precision tends to be low (for any algorithm). The "coverage forgiving" reduced average (C4 in Section 3) for a metric then gives an unfair advantage to kNN compared to MF, as the latter computes recommendations for all users, easy and difficult alike.

Figure 8 shows that as the overlap requirement increases (horizontal axis), kNN surrenders user coverage and, as a consequence, improves in terms of the "forgiving average", while dropping in terms of the full average. (The metric values for MF show as horizontal reference lines in the graphs, since they are unaffected by the kNN configuration.) If a parameter greater than 10 is chosen in MovieLens or greater than 20 in Netflix, this particular experiment "confirms" that kNN is superior to MF. But this comparison would be unfair, since the two methods are being compared over different sets of users, with a selection bias evident in the subset used for the kNN measurement.

Indeed, if the importance of comparing systems on the same set of users is overlooked, a further conclusion from this experiment would be that a high overlap requirement is desirable for kNN. But in truth no real improvement is achieved by doing that, with user coverage being degraded without recommendation gain arising.

Decision D: System Comparison and Statistical Tests

When systems are being compared via their mean scores, claims of "improvement" should be backed up by statistical confidence computations, in both RS evaluation and in IR evaluation (Sakai, 2016, 2018). Most researchers are aware of this expectation. Moreover, given the large number of data points involved in typical RS evaluations (Table 1), it is unusual for an experiment to lack sufficient statistical power, and hence even quite small differences in system performance can be reliably detected.

Returning to the starting configuration, the first two rows of Table 3 give p values for a two-tailed paired t-test, comparing the kNN and MF approaches using Prec@10 and NDCG@10 on full rankings. The remainder of that table explores other metric/gain functions. Table 4 also reports significance, noting situations in which $p < 0.05$ was detected. With the exception of just three combinations – NDCG@10 applied to full rankings, RBP with the binarized gain mapping $g_1(\cdot)$, and test-weighted aggregation when coupled with Prec@10 – a researcher who carried out any of these evaluations would, in isolation, conclude that they had attained a significant result.

The problem is, in some of these many experiments it is kNN that is significantly better, and in others it is MF. Of course, there is no particular reason to expect that if two different metrics are used in an evaluation, or even if two different evaluation depths are used, that both should give the same outcome. But if such deceptively simple changes can give rise to opposing statistically significant outcomes, it does mean that all aspects of the comparison need to be carefully considered, because it might be that the improvement that is being claimed is less a consequence of innate superiority, and more a quirk of the particular methodology that was followed.

5 Summary and Conclusions

We have presented a taxonomy of the methodological decisions that are required during the design of an RS experiment, and discussed the choices that are available at each of a total of eight distinct steps. We have also carried out a range of experimentation using two public RS implementations and two public datasets, to show how different experimental choices can affect the outcome of an experiment.

Some of these outcomes will come as no surprise to researchers. Changing the evaluation metric, for example, clearly results in a different experiment being undertaken, and means that a different outcome might result. But some of the other outcomes should probably be of concern to the community. For example, choosing a different split ratio when setting up Train and Test can affect the outcome, even when all other factors are held constant; as can altering the depth of evaluation of the (same) metric; as can the choice of gain function; as can shifting from user-weighted aggregation to volume-weighted aggregation.

The critical point that we seek to make with this work is this: if experimental outcomes can be affected by the methodological decisions made in these regards just as much as they are by the innate quality of the RS systems being compared, then great care is required when carrying out evaluations. The very first step should be a detailed statement in regard to experimental settings. That is, even if the sole benefit of this paper is to enumerate a taxonomy against which others can report their experimental design, we feel we will have made an important contribution.

A fundamental question that we have not sought to answer, and is perhaps unanswerable, is the choice (C2) between full rankings and condensed rankings. The benefit of full rankings is that they reflect what it is that is presented to the user of an operational system; the benefit of condensed rankings is that they reflect what it is that we can properly measure using the standard batch-oriented Train/Test methodology. Neither approach seems truly satisfactory, because the advantage of each is the Achilles' heel of the other. We urge experimental researchers to remain keenly aware of these limitations. Perhaps a third approach will emerge as a result of future developments in the area, or perhaps it will become commonplace for both results to be presented in future evaluations. What is clear is that user-based evaluation via a continuous stream of user interaction data in response to the recommendations flowing from a live system is preferable to static batch-oriented approaches in this regard. But not all researchers have access to such systems and their user behavior data, and it seems likely that batch-oriented evaluation will continue to play an important role.

The reader might feel that we "got lucky" with this investigation, because the split ratio of 0.5 that we happened to choose for the starting configuration highlighted many of the issues we sought to discuss. We do not, however, believe that our "luck" in this regard detracts from the message that we seek to convey – if anything, it makes it stronger, since our "good luck" is also "bad

luck" from the opposite point of view. A researcher who (for whatever reason) worked with the MovieLens dataset and chose a split ratio of 0.4 (perhaps because they wished to have deeper evaluation of Test) or a split ratio of 0.8 (perhaps because they planned to carry out five-fold cross-validation) might be misled by their initial findings, and discouraged from pursuing what might actually be a profitable line of enquiry. Likewise, the existence of a statistically significant result arising from some particular combination of methodological decisions might well sway a referee, and support them in making an "accept" recommendation for a paper without them appreciating that the comparison that led to the particular p value might be fragile. Armstrong et al. (2009) make somewhat similar observations in regard to IR evaluation.

We have not presented any new RS techniques in this work; that is not our objective. Nor have we compared and tested a broad range of, or the most-recent, RS mechanisms, and nor have we explored a wide range of datasets; exhaustive coverage of methods or test suites is also not our objective. Rather, our purpose with this work – via two standard implementations and two standard datasets – is to advise and request that researchers and practitioners be alert to the many things that might affect the outcomes of the experiments that they carry out. In particular, we ask that researchers carefully document the options they employed, perhaps using our taxonomy or one that is even more detailed, so that others can hope to reproduce the same experimental set-up; and that they likewise hesitate before making what might turn out to be overly-bold claims based on single experimental configurations. Just as we got "lucky" with the arrangements that we chose to work with to demonstrate the possible pitfalls, so too might a researcher get "lucky" (or "unlucky") with the test arrangements they apply when measuring some proposed new technique.

Our findings are somewhat sobering. The 2019 tabulation and testing of recent RS techniques by Dacrema et al. (2019) documents one way in which RS researchers need to be careful (that is, in their choice of baseline systems against which to compare); our work here adds to those concerns, by exposing other ways in which experiments must be planned strategically. Neither of these two different aspects of experimental methodology can be neglected if the field is to progress.

We close by reiterating our primary observation: methodological choices *can* and *do* affect experimental outcomes, and researchers are encouraged to fully document all of the decisions they make when carrying out RS experiments, so that others seeking to build on those findings have a firm basis on which to do so. We also encourage reporting of multiple methodological combinations, so that any quirks and idiosyncrasies associated with particular settings can be recognized for what they are. If a new system can be shown to be better than a suite of good baselines over a range of parameter settings and experimental configurations, then it probably is a superior system. But if the new systems turns out to only be better in one configuration, or using one data set, or with one metric, then claims in regard to its superiority need to be approached with scepticism and caution.

On a more positive note, we are confident that the majority of experienced RS researchers and practitioners are aware of the need for wide-ranging experimentation; and the discussion here should not be taken as criticism of the community, but rather, as a guide to people new to the field that explains issues that they must, of necessity, be aware of and alert to.

Acknowledgment

The first two authors were funded in part by grant TIN2016-80630-P from the Spanish Ministry of Science, Innovation and Universities.

References

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans on Knowledge and Data Engineering 17(6):734–749

Armstrong TG, Moffat A, Webber W, Zobel J (2009) Improvements that don't add up: Ad-hoc retrieval results since 1998. In: Proc. ACM Int. Conf. on Information and Knowledge Management (CIKM), pp 601–610

Bailey P, Craswell N, Soboroff I, Thomas P, de Vries AP, Yilmaz E (2008) Relevance assessment: Are judges exchangeable and does it matter. In: Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR), pp 667–674

Bellogín A, Castells P, Cantador I (2011) Precision-oriented evaluation of recommender systems: An algorithmic comparison. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp 333–336

Bellogín A, Wang J, Castells P (2013) Bridging memory-based collaborative filtering and text retrieval. Information Retrieval 16(6):697–724

Bellogín A, Castells P, Cantador I (2017) Statistical biases in information retrieval metrics for recommender systems. Information Retrieval 20(6):606–634

Bertin-Mahieux T, Ellis DPW, Whitman B, Lamere P (2011) The million song dataset. In: Proc. Int. Soc. for Music Information Retrieval Conf. (ISMIR), pp 591–596

Cañamares R, Castells P (2017) A probabilistic reformulation of memory-based collaborative filtering: Implications on popularity biases. In: Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR), pp 215–224

Cañamares R, Castells P (2018a) Characterization of fair experiments for recommender system evaluation: A formal analysis. In: RecSys Wrkshp. on Offline Evaluation of Recommender Systems (REVEAL)

Cañamares R, Castells P (2018b) Should I follow the crowd? A probabilistic analysis of the effectiveness of popularity in recommender systems. In: Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR), pp 415–424

Carterette BA (2012) Multiple testing in statistical analysis of systems-based information retrieval experiments. ACM Trans on Information Systems 30(1):4:1–4:34

Castells P, Hurley NJ, Vargas S (2015) Novelty and diversity in recommender systems. In: Recommender Systems Handbook, Springer, pp 881–918

Chapelle O, Metlzer D, Zhang Y, Grinspan P (2009) Expected reciprocal rank for graded relevance. In: Proc. ACM Int. Conf. on Information and Knowledge Management (CIKM), pp 621–630

Cremonesi P, Koren Y, Turrin R (2010) Performance of recommender algorithms on top-n recommendation tasks. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp 39–46

Dacrema MF, Cremonesi P, Jannach D (2019) Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp 101–109

Dinçer BT, Macdonald C, Ounis I (2014a) Hypothesis testing for the risk-sensitive evaluation of retrieval systems. In: Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR), pp 23–32

Dinçer BT, Ounis I, Macdonald C (2014b) Tackling biased baselines in the risk-sensitive evaluation of retrieval systems. In: Proc. European Conf. on Information Retrieval (ECIR), pp 26–38

Ferro N, Fuhr N, Grefenstette G, Konstan JA, Castells P, Daly EM, Declerck T, Ekstrand MD, Geyer W, Gonzalo J, Kuflik T, Lindén K, Magnini B, Nie JY, Perego R, Shapira B, Soboroff I, Tintarev N, Verspoor K, Willemsen MC, Zobel J (2018) From evaluating to forecasting performance: How to turn Information Retrieval, Natural Language Processing and Recommender Systems into predictive sciences. Dagstuhl Manifestos 7(1):96–139

Gilotte A, Calauzènes C, Nedelec T, Abraham A, Dollé S (2018) Offline A/B testing for recommender systems. In: Proc. ACM Int. Conf. on Web Search and Data Mining (WSDM), pp 198–206

Gruson A, Chandar P, Charbuillet C, McInerney J, Hansen S, Tardieu D, Carterette B (2019) Offline evaluation to make decisions about playlist recommendation. In: Proc. ACM Int. Conf. on Web Search and Data Mining (WSDM), pp 420–428

Gunawardana A, Shani G (2015) Evaluating recommendation systems. In: Recommender Systems Handbook, Springer, pp 265–308

Harman DK (2005) The TREC test collections. In: Voorhees EM, Harman DK (eds) TREC: Experiment and Evaluation in Information Retrieval, MIT Press, chap 2, pp 21–52

Harper FM, Konstan JA (2016) The MovieLens datasets: History and context. ACM Trans on Interactive Intelligent Systems 5(4):19.1–19.19

He R, McAuley J (2016) Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proc. Int. Conf. on the World Wide Web (WWW), pp 507–517

Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans on Information Systems 22(1):5–53

Hofmann K, Li L, Radlinski F (2016) Online evaluation for information retrieval. Foundations & Trends in Information Retrieval 10(1):1–117

Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Proc. Int. Conf. on Data Mining (ICDM), pp 15–19

Jannach D, Lerche L, Kamehkhosh I, Jugovac M (2015) What recommenders recommend: An analysis of recommendation biases and possible countermeasures. User Modeling and User-Adapted Interaction 25(5):427–491

Järvelin K, Kekäläinen J (2002) Cumulated gain-based evaluation of IR techniques. ACM Trans on Information Systems 20(4):422–446

Kazai G, Kamps J, Milic-Frayling N (2013) An analysis of human factors and label accuracy in crowdsourcing relevance judgments. Information Retrieval 16(2):138–178

Kutlu M, McDonnell T, Barkallah Y, Elsayed T, Lease M (2018) Crowd vs. expert: What can relevance judgment rationales teach us about assessor disagreement? In: Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR), pp 805–814

Lu X, Moffat A, Culpepper JS (2016) The effect of pooling and evaluation depth on IR metrics. Information Retrieval 19(4):416–445

Marlin BM, Zemel RS (2009) Collaborative prediction and ranking with non-random missing data. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp 5–12

Moffat A, Zobel J (2008) Rank-biased precision for measurement of retrieval effectiveness. ACM Trans on Information Systems 27(1):2.1–2.27

Moffat A, Bailey P, Scholer F, Thomas P (2017) Incorporating user expectations and behavior into the measurement of search effectiveness. ACM Trans on Information Systems 35(3):24:1–24:38

Ning X, Desrosiers C, Karypis G (2015) A comprehensive survey of neighborhood-based recommendation methods. In: Recommender Systems Handbook, Springer, pp 37–76

Ricci F, Rokach L, Shapira B (2015) Recommender systems: Introduction and challenges. In: Recommender Systems Handbook, Springer, pp 1–34

Robertson S (2006) On GMAP: And other transformations. In: Proc. ACM Int. Conf. on Information and Knowledge Management (CIKM), pp 78–83

Said A, Bellogín A (2014) Comparative recommender system evaluation: Benchmarking recommendation frameworks. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp 129–136

Sakai T (2016) Statistical significance, power, and sample sizes: A systematic review of SIGIR and TOIS, 2006-2015. In: Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR), pp 5–14

Sakai T (2018) Laboratory Experiments in Information Retrieval: Sample Sizes, Effect Sizes, and Statistical Power, The Information Retrieval Series, vol 40. Springer

Sakai T, Kando N (2008) On information retrieval metrics designed for evaluation with incomplete relevance assessments. Information Retrieval 11(5):447–470

Schnabel T, Swaminathan A, Singh A, Chandak N, Joachims T (2016) Recommendations as treatments: Debiasing learning and evaluation. In: Proc. Int. Conf. on Machine Learning (ICML), pp 1670–1679

Steck H (2010) Training and testing of recommender systems on data missing not at random. In: Proc. Conf. on Knowledge Discovery and Data Mining (KDD), pp 713–722

Steck H (2011) Item popularity and recommendation accuracy. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp 125–132

Steck H (2013) Evaluation of recommendations: Rating-prediction and ranking. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp 213–220

Swaminathan A, Krishnamurthy A, Agarwal A, Dudík M, Langford J, Jose D, Zitouni I (2017) Off-policy evaluation for slate recommendation. In: Proc. Conf. on Neural Information Processing Systems (NIPS), pp 3635–3645

Valcarce D, Bellogín A, Parapar J, Castells P (2018) On the robustness and discriminative power of IR metrics for top-n recommendation. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp 260–268

Wasserstein RL, Schirm AL, Lazar NA (2019) Moving to a world beyond "$p < 0.05$". The American Statistician 73(sup1):1–19

Webber W, Moffat A, Zobel J (2008) Score standardization for inter-collection comparison of retrieval systems. In: Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR), pp 51–58

Webber W, Moffat A, Zobel J (2010) A similarity measure for indefinite rankings. ACM Trans on Information Systems 28(4):20.1–20.38

Yang L, Cui Y, Xuan Y, Wang C, Belongie S, Estrin D (2018) Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In: Proc. ACM Conf. on Recommender Systems (RecSys), pp 279–287

Yilmaz E, Aslam JA, Robertson S (2008) A new rank correlation coefficient for information retrieval. In: Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR), pp 587–594