Chapter 9

Offline and online evaluation of recommendations

Alejandro Bellogín[†], Alan Said[‡]

[†] *Universidad Autónoma de Madrid, Madrid, Spain.*
alejandro.bellogin@uam.es
[‡] *University of Skövde, Skövde, Sweden.*
alansaid@acm.org

Recommender Systems have been a popular research topic within personalized systems and information retrieval since the mid nineties. Throughout this time, various models of recommendation have been developed, e.g., approaches using collaborative filtering for purposes such as retrieval of ranked lists of items for consumption, or for the rating prediction task (which was made very popular through the Netflix prize). Today, the use of recommender systems has spread to a very wide area of topics, including personalized healthcare, online news portals, food, social networks, exercise, jobs, investment, transportation, shopping, etc. Given the various situations recommendations can be applied to, it follows that evaluation of these systems needs to be tailored to the specific setting, domain, user-base, context, etc. This chapter aims to give an overview of some of the more commonly used evaluation methods and metrics used for various types of recommendation techniques. We also provide a summary of the available resources in this topic, in addition to some practical considerations, experimental results, and future directions about evaluation in recommendation.

9.1 Introduction

The evaluation of Recommender Systems (RS) has been, and still is, the subject of active research in the field, where open questions remain [Herlocker *et al.* (2004); Gunawardana and Shani (2015)]. Since the advent of the first RS, recommendation performance has been usually equated to the accuracy of rating prediction, that is, estimated ratings are compared against actual ratings, and differences between them are computed by means of error-based metrics such as the Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE). In terms of the effective utility of recommendations for users, there is however an increasing realization that the quality (precision) of a ranking of recommended items can be more important than the accuracy in predicting specific rating values [Knijnenburg and Willemsen (2015)]. As a result, precision-oriented metrics are being increasingly considered in the field, and a large amount of recent work has focused on evaluating top-N ranked recommendation lists with the above type of metrics. Precision in this context can be interpreted as the relevance of the recommended item, i.e. the likelihood of the item being watched, liked, or otherwise consumed by the user.

Nonetheless, the recent realization that high prediction accuracy might not translate to a higher perceived appreciation from the users has brought a plethora of novel metrics and methods, focusing on other aspects of recommendation [Said *et al.* (2013a); Castells *et al.* (2015)]. More specifically, other dimensions apart from accuracy – such as coverage, diversity, novelty, and serendipity – have been recently taken into account and analyzed when considered what makes a good recommendation [Said *et al.* (2014b); Cremonesi *et al.* (2011); McNee *et al.* (2006); Bollen *et al.* (2010); Mesas and Bellogín (2017)].

With this in mind comes the understanding that evaluation is the key to identifying how well an algorithm or a system works. Deploying a new algorithm in a new system will have an effect on the overall performance of the system – in terms of accuracy and other types of metrics. Both prior deploying the algorithm, and after the deployment, it is important to evaluate the system performance. However, the evaluation strategies, metrics, and methodologies need to consider that the use of RSs has spread to a very wide area of topics, including personalized healthcare [Elsweiler *et al.* (2015); Luo *et al.* (2016)], online news portals [Said *et al.* (2014a)], food [Elahi *et al.* (2015, 2014)], social networks [Guy (2015)], exercise [Berkovsky *et al.* (2012)], jobs [Abel (2015)], investment [Zhao *et al.* (2015)], transportation [Bistaffa *et al.* (2015)], shopping [Jannach *et al.* (2015a)], etc. and adapt specifically to each use-case.

The rest of the chapter is organized as follows. Section 9.1.1 defines basic concepts used when evaluating recommender systems, then, Sec. 9.1.2 and

Sec. 9.1.3 present in more detail the specifics of offline and online evaluation, and Sec. 9.2 provides some algorithmic solutions specifically devoted for evaluation in the areas described in the first part of this book. Then, Sec. 9.3 shows resources – such as datasets and libraries – currently available to perform RS evaluation. Section 9.4 presents experimental results where we show how the different evaluation methodologies and metrics compare against each other. And, finally, in Sec. 9.5 and Sec. 9.6 we discuss some practical considerations about the problem of RS evaluation, and include future directions worth of further involvement from the community.

9.1.1 *Basic Concepts in Evaluation*

The evaluation of RSs has been a major object of study in the field since its earliest days, and it is still a topic of ongoing research, where open questions remain [Herlocker *et al.* (2004); Gunawardana and Shani (2015)]. It is acknowledged that the evaluation of RSs should take into account the goal of the system itself. For example, [Herlocker *et al.* (2004)] identify two main user tasks: *annotation in context* and *find good items*. In these tasks the users only care about errors in the item rank order provided by the system, not the predicted rating value itself. Based on this consideration, researchers have started to use precision-based metrics to evaluate recommendations – as we shall see later – although most works also still report error-based metrics for comparison with state of the art approaches. Moreover, other authors [Herlocker *et al.* (2004); Gunawardana and Shani (2015)] encourage considering alternative performance criteria, like the novelty of the suggested items and the item coverage of a recommendation method. Throughout this chapter we describe these types of evaluation metrics.

However, not all the evaluation metrics can be computed under any experimental settings. Different evaluation protocols exist and they impose constraints on the type of data that can be measured and analyzed. Two main evaluation protocols are usually considered: offline and online evaluation. These protocols present a clear tradeoff between effort (time, users, etc.) and usefulness/trustworthiness of the results, which will be discussed in the subsequent sections. We will also briefly introduce how user studies fit in this context, a protocol with some advantages and disadvantages of the two previously mentioned evaluation protocols.

9.1.2 *Offline evaluation*

Offline evaluation allows to compare a wide range of candidate algorithms at a low cost, it is easy to conduct and does not require any interaction with real users. However, user studies and online experiments are more trustworthy – since the system is used by real users and interacted with in real time – but care must be taken to consider biases in the experimental design [Gunawardana and Shani (2015)].

An important decision in the experimental configuration of RS evaluation is the dataset partitioning strategy. How the datasets are partitioned into training and test sets may have a considerable impact on the final performance results, and may cause some recommenders to obtain better or worse results depending on how the partitioning process is configured.

There are several choices to be made when considering offline evaluation, first, we should choose whether or not to take time into account [Gunawardana and Shani (2015)]. Time-based approaches require that user interaction records have timestamps. A simple approach is to select a time point in the available interaction data timeline, and to separate the data at that point. The resulting data subsets can then be used as training data (all interaction records prior to the time point) and test data (all interaction dated after the split time point), a simple example is shown in Figure 9.1 (right). The split point can be set so as to, for instance, have a desired training/test ratio in the experiment or defining a specific time window for the training and test splits [Campos *et al.* (2014)]. The ratio between training and test data can be global, with a single common split point for all users, or user-specific, to ensure the same ratio per user. Time-based approaches have the advantage of more realistically matching working application scenarios, where "'future"' user likes (which would translate to positive response to recommendations by the system) are to be predicted based on past evidence.

In the case when time is not a factor, there are at least the following three strategies to select which items are selected into the training data and which are selected into the test data correspondingly. These are: a) sample a fixed number (different) for each user; b) sample a fixed (but the same for all) number for each user, also known as *given n* or *all but n* protocols; c) sample a percentage of all the interactions using cross-validation. Commonly, the last protocol is used [Goldberg *et al.* (2001)], although several authors have also used the *all but n* protocol [Breese *et al.* (1998)]. Figure 9.1 (left) shows an example of a random dataset partition

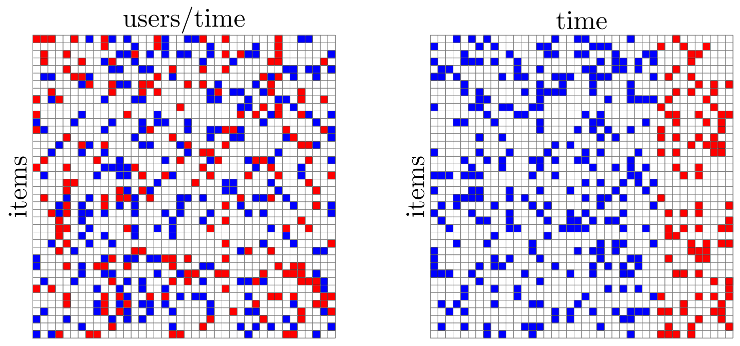


Fig. 9.1: How the dataset would be split into training (blue) and test (red) sets, for a random (left) and temporal (right) split. White cells denote unknown values in the user-item matrix.

not taking time into consideration.

Recently, some researchers have started to prune the datasets by creating p-cores, where every user and item has at least p interactions [Jäschke *et al.* (2007)], in order to reduce the inherent sparsity existing in RS datasets. Nonetheless, independently from the dataset partitioning, it is recognized that the goals for which an evaluation is performed may be different for each situation, and thus, a different setting (and partitioning protocol) should be developed [Herlocker *et al.* (2004); Gunawardana and Shani (2015)]. If that is not the case, the results obtained in a particular setting would be biased and difficult to use in further experiments, for instance, in an online experiment.

Regarding the actual evaluation process, there is a relation between the evaluation protocol and the evaluation metrics that can be computed. Error metrics require explicit ground truth values for every evaluated user-item pair – that is, only items in the test set of each user will be considered. Ranked recommendations (using the metrics mentioned before), on the other hand, require for a target user u to select two sets of items, namely relevant and not relevant items. The following candidate generation strategies, where L_u denotes the set of target items the recommender ranks (candidate items), have been proposed (we follow the notation presented in [Said and Bellogín (2014)]):

UserTest (UT) This strategy takes the same target item sets as standard error-based evaluation: for each user u, the list L_u consists of items rated by u in the test set. The smallest set of target items for each user is selected, including no unrated items. A relevance threshold is used to indicate which of the items in the user's test are considered relevant. Threshold variations can be static for all users [Jambor and Wang (2010)], or per-user [Basu *et al.* (1998)].

TrainItems (TI) Every rated item in the system is selected – except those rated by the target user. This strategy is useful when simulating a real system where no test is available, i.e. no need to look into the test set to generate the rankings [Bellogín *et al.* (2011)]. The relevant items for each user consist of those included in her test set; the use of a threshold to consider only highly rated items is optional although recommended.

RelPlusN (RPN) For each user, a set of highly relevant items is selected from the test set. Then, a set of non-relevant items is created by randomly selecting N additional items. In [Cremonesi *et al.* (2010)], N is set to $1,000$ stating that the non-relevant items are selected from items in the test set not rated by u. Finally, for each highly relevant item i, the recommender produces a ranking of the union between this item and the non-relevant items.

As it was observed in [Bellogín *et al.* (2011)] and [Said and Bellogín (2014)], each of these candidate generation strategies may produce a different ranking of recommendation performance – TrainItems and RelPlusN produce consistent results (although with different absolute values) whereas UserTest obtains results closer to those from error-based metrics. Hence, we should pay attention to the strategy used when ranking metrics are computed, since the amount of relevant items considered can drastically change the output of the experiment. In contrast to other fields such Information Retrieval (IR), in RS we have to define training and test sets, whereas in IR, we would have the whole dataset available, first, for the indexing task, and then, for the retrieval and evaluation tasks. In RS, we need to separate the data into training and test; the more training available, the better the algorithm will learn the users' preferences. However, the smaller the test set, the smaller the confidence on the obtained results. This sparsity in the ground truth dimension may produce biases in the evaluation results, as observed in [Bellogín *et al.* (2017)].

Once the splitting and evaluation methodology are decided, we can estimate the performance of the system by computing different evaluation metrics on the results reported by the recommendation algorithm. As mentioned before, depending on the selected methodology, it might not be possible to compute some metrics. More importantly, depending on the algorithm being tested some of the evaluation metrics that we are going to present now cannot be applied.

In the classical formulation of the recommendation problem, user preferences for items are represented as numeric ratings, and the goal of a recommendation algorithm consists of predicting unknown ratings based on known ratings and, in some cases, additional information about users, items, and the context. In this scenario, the accuracy of recommendations has been commonly evaluated by measuring the error between predicted and known ratings, using **error metrics**. Traditionally, the most popular metrics to measure the accuracy of a RS have been the Mean Absolute Error (MAE), and the Root Mean Squared Error (RMSE):

$$\text{MAE} = \frac{1}{|\text{Te}|} \sum_{(u,i)\in\text{Te}} |\tilde{r}(u,i) - r(u,i)| \tag{9.1}$$

$$\text{RMSE} = \sqrt{\frac{1}{|\text{Te}|} \sum_{(u,i)\in\text{Te}} (\tilde{r}(u,i) - r(u,i))^2} \tag{9.2}$$

where $\tilde{r}$ and r denote the predicted and real rating, respectively, and Te corresponds to the test set. The RMSE metric is usually preferred to MAE because it penalizes larger errors.

A critical limitation of these metrics is that they do not make any distinction between the errors made on the top items predicted by a system, and the errors made for the rest of the items. Furthermore, they can only be applied when the recommender predicts a score in the allowed range of rating values. Because of that, implicit and some content-based and probabilistic recommenders cannot be evaluated in this way, since $\tilde{r}(u,i)$ would represent a probability or, in general, a preference score, not a rating.

Although dominant in the literature, some authors have argued that the error-based evaluation methodology is detrimental to the field since the recommendations obtained in this way are not the most useful for users [McNee *et al.* (2006)]. Acknowledging this, recent work has evaluated top-N ranked recommendation lists with **precision-oriented metrics** [Cremonesi *et al.* (2010); McLaughlin and Herlocker (2004); Bellogín *et al.* (2011)], drawing from well studied evaluation methodologies in the IR field.

Among the wide range of precision-oriented metrics based on rankings, the most typical ones are precision, recall, normalized discounted cumulative gain, mean average precision, and mean reciprocal rank. Each of these metrics captures the quality of a ranking from a slightly different angle. More specifically, precision accounts for the fraction of recommended items that are relevant, whereas recall is the fraction of the relevant items that has been recommended. Both metrics are inversely related, since an improvement in recall typically produces a decrease in precision. They are typically computed up to a ranking position or cutoff k, being denoted as $P@k$ and $R@k$ [Baeza-Yates and Ribeiro-Neto (2011)]. Note that recall has also been referred to as hit-rate in [Deshpande and Karypis (2004)]. Hit-rate has also been defined as the percentage of users with at least one correct recommendation [Bellogín *et al.* (2013)], corresponding to the success metric (or first relevant score), as defined by TREC [Tomlinson (2012)].

The mean average precision (MAP) metric provides a single summary of the user's ranking by averaging the precision figures obtained after each new relevant item is obtained [Baeza-Yates and Ribeiro-Neto (2011)]. Normalized discounted cumulative gain (nDCG) uses graded relevance that is accumulated starting at the top of the ranking and may be reduced, or discounted, at lower ranks [Järvelin and Kekäläinen (2002)]. Using a different discount function, the rank score or half-life utility metric [Breese *et al.* (1998); Herlocker *et al.* (2004)] can be obtained from the nDCG formulation. Mean reciprocal rank (MRR) favors rankings whose first correct result occurs near the top ranking results [Baeza-Yates and Ribeiro-Neto (2011)]. This metric is similar to the average rank of correct recommendation (ARC) proposed in [Burke (2004)] and to the average reciprocal hit-rank (ARHR) defined in [Deshpande and Karypis (2004)].

In a more modern formulation of the recommendation problem, the ratings are no longer important. Instead, the consumption of recommended items by users is key, i.e., whether a recommended movie will be seen or a music track listened to. In this context, it is important to ask oneself what the recommender system should bring the user. If a recommendation algorithm suggests an item that the user is already aware of, what is the value of the system? Will this recommendation result in the consumption of the item? The common assumption is that a recommender system, in this context, should bring the user something she might not yet be familiar with, i.e., something novel, unexpected, or serendipitous. Still, the novel, unexpected, or serendipitous recommendations need to fulfill the requirement of the items being of actual interest to the user. These, so-called, non-accuracy metrics focus on the variety, popularity, novelty and similar aspects of the items, or lists of items, that are recommended [Castells *et al.* (2015)].

Due to the nature of non-accuracy metrics, there are often various definitions of them, each tailored towards the context they are used in. Furthermore, it is difficult to create a ground truth dataset to use with these metrics. Hence, recommendation algorithms which are specifically tailored towards non-accuracy metrics will often perform badly in terms of accuracy-based metrics [Said *et al.* (2013a)]; and symmetrically, if an algorithm is tailored towards accuracy metrics, it will often perform badly in terms of non-accuracy metrics.

Perhaps the most well-known non-accuracy metric, serendipity, attempts to model what is often referenced to as a *pleasant and unexpected surprise*. Serendipity is a compound metric of, among others, novelty and diversity. Novelty expresses how new a recommended item is (for a user). The underlying motivation for novelty being an interesting aspect of recommendation is that items which are old, or rather not new, can already have been seen by the user. If this is the case, recommending items which are already known by the user might not be of much value, since the recommendation does not actually present the user with something she could not find herself. Novelty can be directly measured in online experiments by asking users whether they are familiar with the recommended item [Celma and Herrera (2008)]. However, it is also interesting to measure novelty in an offline experiment, so as not to restrict its evaluation to costly and hard to reproduce online experiments. While novelty often can have a negative effect on accuracy, diversity can often be increased without necessarily sacrificing accuracy. Diversity expresses, as implied, the variety of the recommended items. There are multiple ways of measuring diversity in a set of recommended items [Castells *et al.* (2015)], commonly this is done by measuring the intra-list diversity (ILD) [Smyth and McClave (2001)] which is defined as

$$\mathrm{ILD} = \frac{1}{|R|(|R|-1)} \sum_{i \in R} \sum_{j \in R} (1 - s(i,j)) \tag{9.3}$$

where R is the list of recommended items and $s(i,j)$ is a similarity measure reporting on the similarity of items i and j given some predefined set of item features. In essence, what ILD calculates is the aggregate diversity of the list of recommended items, i.e. the more similar (opposite of diverse) the items in the list are (given the selected similarity measure), the lower diversity will the list have. When using ILD as a measure, the goal is to generate a list of recommended items that contains items that are both accurate and diverse.

Furthermore, an often forgotten dimension of evaluation, at least in academic research, are those metrics related to the development, and maintenance of the recommender system itself, such as CPU cost per recommendation, storage cost, cost of re-training the model, etc. The coverage of the list of recommended items – i.e., how well does the list correspond to what is currently in stock or elseways available – also fits in this dimension, and it can be applied not only to the catalog of available items but to the users, hence, an algorithm which can recommend very accurate items, but only for a small portion of users might not be as "good" as a slightly less accurate algorithm with a higher user coverage. In [Gunawardana and Shani (2015)] two metrics are proposed for measuring item coverage: one based on the Gini index, and another based on Shannon's entropy. In [Ge *et al.* (2010)] the authors propose simple ratio quantities to measure such metrics, and to discriminate between the percentage of the items for which the system is able to generate a recommendation (prediction coverage), and the percentage of the available items that are effectively ever recommended (catalog coverage).

Finally, a last step once the results from the evaluation metrics have been obtained is to perform some type of statistical testing. There exist different options to check for significance on the results or on the difference between a method and a baseline: paired/unpaired tests, effect size, confidence intervals, etc. [Sakai (2014)]. It is important to be as specific as possible regarding which procedure was followed and the method used; additionally, to facilitate the interpretation of the results, related statistics such as the mean, variance, and population size of the samples should also be reported.

More importantly, when performing any type of statistical testing method, the data on which the method was computed must be specified, since, as with other aspects of the recommendation process, there is no standard procedure yet, especially when running cross-validated splits, where more than one test split is used and, hence, more than one performance measurement is obtained, which could lead to inconsistent conclusions about the significance of the results if performed in an split basis, e.g., a significant difference is found in some folds but not in others. On the other hand, if the results from each split (on a user basis, as it is done in IR for queries) are concatenated one after the other, may distort the test, because the data points are not independent (the same user appears more than once) and the number of points increase substantially [Bouckaert (2003); Kosir *et al.* (2013)].

	u_1	u_2	u_3	u_4	u_5
i_1	1	1	0	0	1
i_2	1	0	1	1	1
i_3	0	0	0	1	0
i_4	1	0	1	0	1
i_5	0	0	1	1	0
i_6	0	0	0	1	0

Table 9.1: A user-item matrix divided into a training set (above the dashed line) and a test set (below the dashed line).

9.1.3 *Online evaluation and User studies*

Online evaluation, as opposed to traditional offline evaluation is performed through direct involvement of a system's users in order to establish a *qualitative* assessment of the system's quality *as perceived by the end users.* To illustrate one of the key differences between offline evaluation and online evaluation, consider this top-N recommendation scenario: We have a user-item interaction matrix, as shown in Table 9.1. The table shows a matrix of 5 users and 6 items and their interactions, e.g., a 1 represents an interaction (rating, purchase, etc.), a 0 the lack of such. The training/test split is illustrated by the dashed line. In this case, an offline evaluation will only recognize item i_5 as a true positive recommendation for user u_3 and items i_5 and i_6 for user u_4. Users u_1, u_2 and u_5 will not have any true positive recommendations since they have not interacted with any of the items. The evaluation does not consider that the items might actually be liked by the user, if recommended in a real-world situation. Similarly, the fact that u_3 has interacted with i_5 does not need to imply that the item is a good recommendation.

In order to overcome this deficiency, online evaluation attempts to capture the quality of the recommendation as perceived by the users by analyzing their interaction patterns with the system together with explicitly asking questions [Gunawardana and Shani (2015); Pu *et al.* (2012); Kohavi *et al.* (2009)]. Online evaluation sometimes involves a user study. Users can be made aware or encouraged to participate, or participate unknowingly. In real-life systems, the concept of *A/B testing* is readily used to estimate different algorithms' qualities [Kohavi *et al.* (2009)]. A/B testing involves assigning a subset of a system's users to the algorithm under evaluation. In studies of real life systems, users are usually not made aware of their participation in tests [Kohavi *et al.* (2009)]. The interactions of the users are then analyzed and compared to a baseline.

More elaborate user studies – including questionnaires and other explicitly collected information – serve as an alternative to A/B testing. This type of studies commonly involve asking the users questions throughout, or after, their interaction with the system. In studies like this, the participants are naturally aware of their participation in the study. In order to be able to analyze the results quantitatively, the users are asked to agree or disagree with a question in the form of a statement, or we can collect data that is later analyzed at different granularity levels [Knijnenburg and Willemsen (2015)].

There is no default, quality-related, set of questions to ask when performing a recommender systems user study, instead questions are based on the type of quality that is sought for; whether relating to the concepts mentioned above or to rather technical qualities, e.g., time of recommendation, number of items recommended, etc. This type of user studies need to be meticulously planned and executed. If poorly executed, there is a risk of changing the users' opinions, e.g., through suggestive questions, or excessive workload or time involved in answering the questions. Workload and time-related issues can be mitigated by creating an incentive for the users to fulfill the survey, e.g., raffling off vouchers, prizes, etc. If no incentive is given, the time involved in answering the survey creates a decaying effect on the fraction of users who complete the study. When the users are given an incentive, there is a risk that some users will answer the questions quickly (at random) in order to be eligible for the award. In order to mitigate these effects, the number of questions and work load should be kept relatively low.

9.2 Algorithmic Solutions

9.2.1 *Evaluating collaborative filtering algorithms*

The process of evaluating traditional collaborative filtering recommender systems follows the processes described in Section 9.1.1. Historically speaking, collaborative filtering recommender algorithms have been the de facto standard for recommendation. In this context, it is only natural that most standard evaluation methods and strategies have been built with those in mind.

The process of evaluating collaborative filtering recommendation algorithms is inherently tied to the setting of the recommendation, e.g. whether the recommendation algorithm is intended to predict ratings or whether it is intended to identify relevant items for the users. Knowing this allows for further specification of evaluation settings, i.e. selecting whether to perform the evaluation in an online or offline manner (as discussed in Section 9.1.1). However, regardless of whether online or offline, the most central selection to make in the evaluation is the objective function, i.e. RMSE or MAE in a rating prediction scenario; accuracy or non-accuracy metric in the top-N recommendation scenario, or more complex values such as dwell-time or churn rate in online evaluation settings.

9.2.2 *Evaluating social recommendation algorithms*

Social recommendation algorithms take the end user's social graph or other social information into consideration. As such, the evaluation of social recommendation algorithms can be performed identically to the evaluation of traditional collaborative filtering algorithms (as discussed above), unless the recommendation algorithms tailors to a non-trivial social context, such as mutual recommendation of users (e.g. in dating apps). Let us focus on the mutual recommendation scenario. Consider the social graph exemplified in Figure 9.2. If a recommender were to recommend social connections in this graph not knowing the edges seen in the figure. Identifying node 1 as a recommendation to node 6 could be deemed a false positive knowing that the social connection between the two nodes is unidirectional, i.e. node 1 is connected to node 6 and not the other way around. Instead, if we consider the case of nodes 3 and 4, a correct recommendation would be to recommend both nodes to each other. However a recommendation algorithm might only identify node 3 as a recommendation to node 4 (and not the analog inverse). All of these cases need to be covered by the objective function selected for the purpose of evaluating the social recommendations in this graph. Whereas in the case of evaluating collaborative filtering recommendation algorithms, traditional metrics such as precision and recall could be used verbatim, in the case of evaluation of social recommendation, the metrics need to be adapted (or aggregated) to fit the recommendation context. For a more in-depth overview of this topic, see Chap. **??**.

9.2.3 *Evaluating group recommendation algorithms*

Technically, the evaluation methods and metrics for single user item recommendations mentioned in Section 9.1 can be employed for evaluating

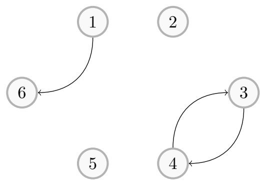


Fig. 9.2: Unidirectional (nodes 1–6) and bidirectional (nodes 3–4) recommendations.

group recommendation algorithms. However, two fundamentally different strategies are common for group recommender evaluation: (i) evaluation of recommendations based on individual user profiles with aggregation of items recommended to each member of a group and (ii) recommendation based on aggregated group profiles. Any common evaluation metric can be used as long as the group is accounted for, by evaluating individual users first, and averaging the evaluation scores for all users in a group, for instance. An example of such an adaptation for the $RMSE$ measure is given in Equation 9.4 where $RMSE_G$ is the $RMSE$ value for the set of users belonging to a group G, where $|G|$ denotes the size of the group.

$$RMSE_G = \frac{1}{|G|} \sum_{u \in G} RMSE_u \qquad (9.4)$$

More details about group recommendation can be seen in Chap. **??**.

9.2.4 *Evaluating context-aware algorithms*

Evaluation of context-aware algorithms requires either a tailored objective function that is able to take into consideration the context of the recommendation or a completely separate evaluation process for each possible instance of our contextual variable. However, again, it is possible to use a traditional evaluation method if certain preparations are done in terms of the dataset. For instance, dividing the users or items into separate contextual profiles, i.e. consider that a user interacts with items on week days and on the weekend. This user could be split up into two separate contextual users, one corresponding to the actions the user has taken on week days, and one corresponding to the weekend activities as exemplified in Table 9.2. Thus, when recommending an item for the weekend profile of said user, only items in the users weekend catalog would be considered true positive recommendations, and, hence, they would affect metrics such as precision or recall.

Table 9.2: An example of how a user-item rating matrix can be contextualized by, e.g. dividing the users into separate profiles for ratings given during weekdays (wd), and ratings given during weekends (we).

(a) Without context.

	u_1	u_2	u_3
i_1	4	4	2
i_2	3	2	
i_3			3

(b) With context.

	u_1^{wd}	u_1^{we}	u_2^{wd}	u_2^{we}	u_3^{wd}	u_3^{we}
i_1	4			4	2	
i_2		3		4		
i_3						3

9.3 Available Resources

In this section, we present some resources related to RS evaluation that are publicly available, including APIs and libraries (Sec. 9.3.1), datasets (Sec. 9.3.2), and competitions (Sec. 9.3.3). We do not consider general tools or environments such as Amazon Mechanical Turk[1] or Crowdflower[2] because they are not specifically tailored to recommender systems experimentation, but they are means to prepare experiments and obtain data from. We do consider, however, frameworks developed in similar areas such as Machine Learning or Information Retrieval where, either explicitly or implicitly, recommendation algorithms or evaluation metrics can be computed or generated.

9.3.1 *APIs and libraries for evaluation*

In general, there are no known APIs that are used to evaluate provided results, mostly because it would involve knowing everything about the recommendation system (users, items, features, etc.). There are, indeed, companies based on serving recommendations as-a-service, but those are also out of the scope of this chapter because the techniques used are generally not disclosed (some examples include BrainSins[3], Criteo[4], and YOOCHOOSE[5]). There are some services based on APIs open to researchers such as plista[6], but since a challenge was organized based on this data, the following section will address this service in detail.

Regarding the software frameworks and libraries, Table 9.3 presents

[1]`https://www.mturk.com/mturk`, accessed October 2017.
[2]`https://www.crowdflower.com`, accessed October 2017.
[3]`https://www.brainsins.com`, accessed October 2017.
[4]`https://www.criteo.com`, accessed October 2017.
[5]`https://yoochoose.com`, accessed October 2017.
[6]`https://www.plista.com`, accessed October 2017.

Table 9.3: An overview of some of the most common open source frameworks used for recommender systems.

Name	**License**	**Language**	**Updated**	**Link**
CARSKit	GPL v2	Java	2017	Source
CofiRank	MPL	C++	2013	Source
Crab	BSD	Python	2012	Website
EasyRec	GPL v2	Java	2016	Website
FastFM	BSD 3	Python	2017	Website
Hi-Rec	MIT	Java	2018	Website
Implicit	MIT	Python	2018	Source
Lenskit	LGPL v2.1	Java	2018	Website
LibFM	GPL v 3	C++	2018	Website
LibRec	GPL v3	Java	2018	Website
LightFM	Apache 2.0	Python	2018	Source
mrec	BSD 3	Python	2016	Source
MyMediaLite	GPL	C# & Java	2017	Website
PREA	BSD	Java	2014	Source
Predictor	MIT	Ruby	2015	Source
Python-recsys	N/A	Python	2014	Source
RankSys	MPL 2.0	Java	2017	Source
RapidMiner	AGPL	Java	2017	Website
RecDB	BSD	PostGreSQL	2018	Source
Recommendable	MIT	Ruby	2018	Source
Recommender 101	Custom	Java	2015	Website
Recommenderlab	GPL v2	R	2017	Website
RecSys.jl	MIT	Julia	2016	Source
RiVal	Apache 2.0	Java	2017	Website
rrecsys	GPL v3	R	2018	Source
SLIM	Other	C	2012	Website
Surprise	BSD 3	Python	2018	Website
SVDFeature	Apache 2.0	C++	2014	Source
TagRec	AGPL 3.0	Java	2018	Source
trec_eval	Other	C	2016	Source
Turi	Apache 2.0	C++	2018	Website
Waffles	LGPL	C++	2018	Source
WrapRec	MIT	C#	2018	Website
QMF	Apache 2.0	C++	2017	Source

a list of the most common open source frameworks that can be used at different stages of the recommendation pipeline. Some of these frameworks/libraries are focused on other fields (Machine Learning, Natural Language Processing, or Information Retrieval) but they provide resources that can be used for recommendation, such as clustering, nearest-neighbors, matrix factorization, etc.

Depending on the application the framework was conceived for, the evaluation techniques and tools could be more or less applicable to recommendation, in particular, when a specific task is aimed. For instance, Machine Learning libraries typically implement error-based metrics such as MAE (see Sec. 9.1.2), whereas Information Retrieval libraries are mostly focused on precision-oriented metrics.

9.3.2 *Datasets for evaluation*

Public datasets including interactions between users and items are of paramount importance in recommender systems research. They serve as input for recommendation algorithms, as simulation data, or for evaluation purposes. Despite their importance, publicly accessible datasets with ratings, clicks, transactions, and so on, have not been abundantly available until very recently, leaving researchers not many options besides classical datasets such as MovieLens, focused on the movie domain.

Table 9.4 shows an heterogeneous list of datasets, including a large number of movie datasets. This might be attributed to historical reasons, since the first public datasets were based on the MovieLens[7] system [Harper and Konstan (2016)] and most of the research developed since has been focused on the movie rating prediction task, neglecting other tasks or domains until recently, when researchers have had access to other, more diverse data sources.

9.3.3 *Competitions about evaluation*

In 2006, one initiative, the Netflix Prize[8], created a focus on recommender systems and contributed to major advancements in the field during its three year run. Similar initiatives have led to great improvements in related fields – e.g., the Text Retrieval Conference[9] in Information Retrieval, and the KDD Cup[10] in Machine Learning and Data Mining communities. Following the success of these, different competitions related to recommendation have appeared, one of them – the RecSys Challenge[11] – organized in conjunction with the ACM Conference on Recommender Systems [Said (2016)].

Throughout the duration of the Netflix Prize, significant advancements were made in the RS research field, e.g., establishing matrix factorization methods such as SVD as state-of-the-art in recommendation. At the 2010 ACM RecSys conference, the seed for what would become the RecSys Challenge was organized as the Challenge on Context-aware Movie Recommendation (CAMRa) [Adomavicius *et al.* (2010)]. CAMRa attracted a moderate number of participants, but contributed to establishing the RecSys Challenge series.

[7]`https://movielens.org`, accessed October 2017.
[8]`http://www.netflixprize.com`
[9]`http://trec.nist.gov`, accessed October 2017.
[10]`http://www.kdd.org/kdd-cup`, accessed October 2017.
[11]`http://www.recsyschallenge.com`

Table 9.4: Datasets commonly used for recommender system evaluation, summarizing the number of users, items, and events (ratings, clicks, or interactions in general) included in the dataset.

Name	**Domain**	**Events**	**Users**	**Items**	**Density**	**Source**
Book-crossing	Books	$1.1 \cdot 10^6$	$2.8 \cdot 10^5$	$2.7 \cdot 10^5$	0.001%	Website
LibimSeTi	Dating	$1.7 \cdot 10^7$	$1.4 \cdot 10^5$	$1.7 \cdot 10^5$	0.076%	Website
Xing (2016)	Jobs	$8.8 \cdot 10^6$	$1.4 \cdot 10^6$	$1.4 \cdot 10^6$	$5 \cdot 10^{-6}$%	Website
Jester	Jokes	$4.1 \cdot 10^6$	$7.3 \cdot 10^4$	10^2	56.34%	Website
Moviepilot (2010)		$4.5 \cdot 10^6$	$1.1 \cdot 10^5$	$2.5 \cdot 10^4$	0.002%	Website
CAMRa2011 (Moviepilot)		$4.4 \cdot 10^6$	$1.7 \cdot 10^5$	$3.0 \cdot 10^4$	0.001%	Website
CAMRa2010 (Filmtipset time)		$5.8 \cdot 10^6$	$3.5 \cdot 10^4$	$5.4 \cdot 10^4$	0.003%	Website
CAMRa2010 (Filmtipset social)		$3.1 \cdot 10^6$	$1.7 \cdot 10^4$	$2.4 \cdot 10^4$	0.008%	Website
Mise-en-scène		$1.3 \cdot 10^7$	$1.8 \cdot 10^5$	$1.3 \cdot 10^4$	$6 \cdot 10^{-9}$%	Website
MovieLens 100k	Movies	$1.9 \cdot 10^5$	10^3	$1.7 \cdot 10^3$	6.30%	Website
MovieLens 1M		10^6	$6.0 \cdot 10^3$	$3.9 \cdot 10^3$	4.25%	Website
MovieLens 10M		10^7	$7.2 \cdot 10^4$	$1.1 \cdot 10^4$	1.31%	Website
MovieLens 20M		$2 \cdot 10^7$	$1.4 \cdot 10^5$	$2.7 \cdot 10^4$	0.05%	Website
MovieLens HetRec		$8.6 \cdot 10^5$	$2.1 \cdot 10^3$	10^4	0.04%	Website
MovieTweetings		$7.0 \cdot 10^5$	$5.3 \cdot 10^4$	$2.6 \cdot 10^4$	0.04%	Website
Netflix		10^8	$4.8 \cdot 10^5$	$1.8 \cdot 10^4$	1.17%	Website
Last.fm 1K		$1.9 \cdot 10^7$	$9.9 \cdot 10^3$	$1.8 \cdot 10^5$	10.91%	Website
Last.fm 360K	Music	$1.7 \cdot 10^7$	$3.6 \cdot 10^5$	$2.9 \cdot 10^5$	0.016%	Website
Last.fm HetRec		$9.3 \cdot 10^4$	$1.9 \cdot 10^3$	$1.8 \cdot 10^4$	0.003%	Website
Yahoo Music (KDD-cup'11)		$2.6 \cdot 10^8$	10^6	$6.2 \cdot 10^5$	0.042%	Website
South Tyrol Suggests	Point of interest	$2.5 \cdot 10^4$	$3.3 \cdot 10^3$	$2.5 \cdot 10^3$	3.1%	Website
Delicious	Tags	$4.2 \cdot 10^8$	10^3	$1.3 \cdot 10^8$	$3 \cdot 10^{-6}$%	Website
Delicious HetRec		$4.4 \cdot 10^5$	$1.8 \cdot 10^3$	$6.9 \cdot 10^4$	0.003%	Website
YOOCHOOSE	Retail	$3.3 \cdot 10^7$	$9.2 \cdot 10^6$	$5.3 \cdot 10^4$	$7 \cdot 10^{-5}$%	Website

The RecSys Challenge has followed a similar structure since its inception: *i)* a dataset and problem are presented, *ii)* teams sign up and participate, *iii)* participants submit their solutions in time for a deadline, *iv)* participants submit papers outlining their approaches, *v)* during a workshop at the ACM RecSys conference participants present their approaches and winners are announced. One of the main features of the challenge is to make available a new real world dataset. This structure is common to other competitions related to RS, such as those presented in Table 9.5.

The RecSys challenge has been constantly evolving to adapt to different trends in the RS community. It has incorporated tasks different to rating prediction and integrating diverse domains – from identifying which groups of users to recommend certain ad campaigns to user engagement prediction in Twitter or e-commerce. Over the years, the challenge has established itself as a benchmarking event for current recommender sys-

Table 9.5: Events and initiatives related to recommender system evaluation.

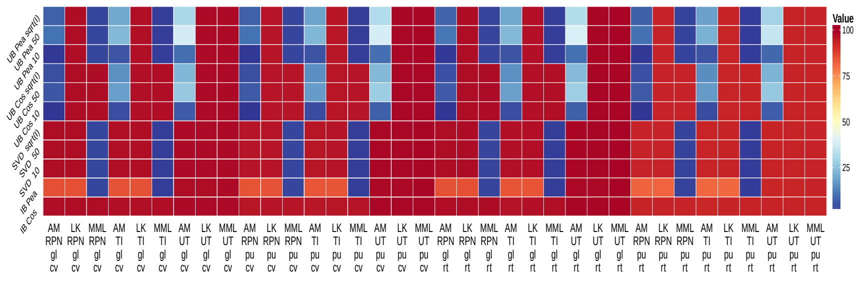
Event	**Occurrence** (first time)	**Link**
ECMLPKDD Discovery Challenge	Not fixed (2008)	Website
Kaggle competitions	Not fixed (2012)	Website
KDD Cup	Not fixed (2007)	Website
NewsREEL	Not fixed (2013)	Website
Netflix Prize	Once (2006)	Website
RecSys Challenge	Yearly (2010)	Website
TREC Contextual Suggestion	Yearly (2012)	Website
WSDM challenge	Not fixed (2018)	Website

tem research. It has attracted participants from academia and industry, allowing researchers and practitioners to learn from, and cooperate with each other, in a community-driven event. Each yearly instance takes on new research challenges based on ongoing trends in industry and academia, which is also evidenced in some of the related events organized in parallel in different venues, such as KDD or WSDM conferences.

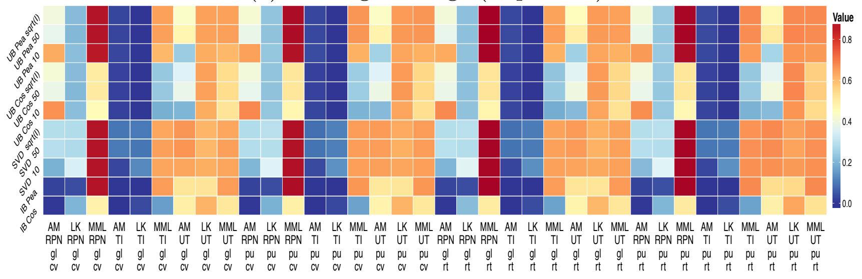
Current and future research in RS acknowledge that certain recommender system settings require online evaluation, i.e., an instantaneous feedback loop between the users of the system and the algorithm. In some instantiations of these competitions, a second stage (if available) usually bring real interactions with users from the system. A special mention deserves the NewsREEL initiative, which allows researchers to receive real recommendation requests by news providers, in the context of the plista recommendation-as-a-service environment. This event started in 2013 as a challenge associated to an ACM RecSys workshop on news recommendation, but has been organized independently since then.

9.4 Experimental Results

A large amount of recommender systems research is based on comparisons of recommendation algorithm's predictive accuracy: the better the evaluation metrics (higher accuracy scores or lower predictive errors), the better the recommender system. However, it is usually difficult to put in context and compare these results, mostly because too many alternatives exist when designing and implementing an evaluation strategy (more on this on Sec. 9.5), and the actual implementation of a recommendation algorithm sometimes diverges considerably from the well-known ideal formulation, frequently due to manual tuning and modifications observed to work better in some situations.



(a) Catalog coverage (in percent).



(b) Normalized discounted cumulative gain at 10 (nDCG@10).

Fig. 9.3: Catalog coverage and nDCG for the controlled evaluation. RPN, TI and UT refer to RelPlusN, TrainItems and UserTest strategies, IB and UB refer to item- and user-based respectively; Pea and Cos to Pearson and Cosine; gl and pu to global and per user; AM (Mahout), LK (LensKit), MML (MyMediaLite) to the frameworks; and cv, rt to cross validation and ratio.

In [Said and Bellogín (2014)], a thorough experimental comparison of different evaluation techniques and recommendation algorithms was presented. In order to achieve comparable evaluation protocols when using different recommendation frameworks and datasets, the authors had complete control of the evaluation dimensions being benchmarked: data splitting, evaluation strategies, and evaluation metrics. The main result found was that there is a large difference in recommendation accuracy across frameworks and strategies, specifically, the same baseline method may perform orders of magnitude better or worse depending on the framework. We include here more details about these results and their corresponding discussion.

Figure 9.3 shows the catalog coverage and nDCG metrics computed using a controlled evaluation protocol. These figures show a wide combination of strategies for data splitting, recommendation and candidate items generation. We notice that, except for the UserTest candidate items strategy, MyMediaLite outperforms Mahout and LensKit in terms of nDCG@10, when using a matrix factorization algorithm (SVD in the figure) and a user-based with Pearson similarity (UB Pea). This high precision comes at the expense of lower coverage, specifically of the catalog (item) coverage. As a consequence, MyMediaLite seems to be able to recommend at least one item per user, but far from the complete set of items, in particular compared to the other frameworks. In terms of nDCG@10, the best results are obtained with the UserTest strategy, with noticeable differences between recommender types, i.e. IB performs poorly, SVD performs well, UB in between, in accordance with, e.g., [Koren and Bell (2015)]. The splitting strategy has little effect on the results in this setting.

Additionally, the results of a framework-dependent evaluation are shown in Table 9.6. For this, we use each framework's internal evaluation classes and report the results obtained. Table 9.6a shows evaluation results in terms of nDCG, generated by Mahout and LensKit, whereas Table 9.6b shows the RMSE values from LensKit and MyMediaLite. We start by studying the results presented in Table 9.6a, where it seems that LensKit outperforms Mahout at several orders of magnitude. The highest nDCG obtained by Mahout (0.2868) is less than one third of the lowest value obtained by LensKit (0.9422). This should be taken in the context of each framework's evaluator. Note that Mahout's evaluator will only consider users with a certain minimum number of preferences (two times the level of recall). Our level of recall was set to 50, meaning only users with at least 100 preferences are evaluated (corresponding only to circa 33% of the users in this dataset).

Looking at the RMSE results obtained by LensKit and MyMediaLite in Table 9.6b, the difference between the frameworks is not as large as in the previous case. All RMSE results are between 7.5% (UBCos50) and 11.7% (UBCos10) of each other. In this case, both frameworks created five instances of training/tests splits with an 80%-20% ratio. The splits are randomly seeded, meaning that even though the frameworks only create five training/test datasets, they are not the same between the two frameworks.

A further observation to be made is how the framework's internal evaluation compares to our controlled evaluation. We see that the frameworks perform better in the controlled environment than in the internal ditto. In the case of nDCG (Table 9.6a), we see that Mahout's values fluctuate more in both versions of the controlled evaluation (RPN and UT) than in Ma-

Table 9.6: Results using the internal evaluation methods of each framework.

(a) nDCG for AM and LK.

Alg.	**F.W.**	**nDCG**
IBCos	AM	0.000414780
	LK	0.942192050
IBPea	AM	0.005169231
	LK	0.924546132
SVD50	AM	0.105427298
	LK	0.943464094
UBCos50	AM	0.169295451
	LK	0.948413562
UBPea50	AM	0.169295451
	LK	0.948413562

(b) RMSE values for LK and MML.

Alg.	**F.W.**	**RMSE**
IBCos	LK	1.01390931
	MML	0.92476162
IBPea	LK	1.05018614
	MML	0.92933246
SVD50	LK	1.01209290
	MML	0.93074012
UBCos50	LK	1.02545490
	MML	0.95358984
UBPea50	LK	1.02545490
	MML	0.93419026

hout's own evaluation. The internal evaluation results consistently remain lower than the corresponding values in the controlled setting – although the RPN values are closer to Mahout's own results. The UT (UserTest) values obtained in the controlled evaluation are several orders of magnitude higher than in the internal setting, even though the setting resembles Mahout's own evaluation closer than the RPN setting. LensKit's internal evaluation consistently shows better results than the controlled setting. We believe this could be an effect related to how the final averaged nDCG metric is calculated, or how the training/test splitting is performed by each framework, where some users might be ignored because of a low number of preferences.

Given the widely differing results, it seems pertinent that in order to perform an inter-framework benchmarking, the evaluation process needs to be clearly defined and both recommendations and evaluations performed in a controlled and transparent environment.

9.5 Practical Considerations

In the light of the previous section, it stands clear that even though different recommendation frameworks implement algorithms in a similar fashion, the results are not comparable, i.e., the performance of an algorithm implemented in one cannot be compared to the performance of the same algorithm in another. Not only do there exist differences in algorithmic implementations, but also in the evaluation methods themselves.

There are no *de facto* rules or standards on how to evaluate a recommendation algorithm. This also applies to how recommendation algorithms of a certain type should be realized (e.g., default parameter values, use of backup recommendation algorithms, and other *ad-hoc* implementations). However, this should perhaps not be seen as something negative *per se.* Yet, when it comes to performance comparison of recommendation algorithms, a standardized (or controlled) evaluation is crucial [Konstan and Adomavicius (2013)]. Without which, the relative performance of two or more algorithms evaluated under different conditions becomes essentially meaningless. In order to objectively and definitively characterize the performance of an algorithm, a controlled evaluation, with a defined evaluation protocol is a prerequisite.

Next, we discuss three main issues that should be considered when implementing and evaluating recommendation algorithms, besides performing a controlled evaluation. First, in Sec. 9.5.1 we describe some typical practical considerations, mostly related to technical aspects and design issues. Then, Sec. 9.5.2 presents a brief analysis about statistical issues that may appear when evaluating recommender systems. Finally, Sec. 9.5.3 summarizes some issues that have an impact on the reproducibility of RS experimental results.

9.5.1 *Design issues to evaluate recommender systems*

One of the first and most obvious design issues that researchers typically assume when evaluating RS is that a user will never consume an item more than once. This results in test splits with no overlap with the training split – as common practice in Machine Learning. Such hypothesis was probably inherited from the most common domain used in the first years by the community: the movie domain; here, the datasets would only contain one interaction (i.e., a rating) once for each user-item pair. This is in contrast with other domains such as music or e-commerce [Schedl *et al.* (2015); Linden *et al.* (2003)], where repeated consumption is paramount and encouraged by the system.

Another constraint (somewhat artificially) imposed in many works when evaluating RS is that of ignoring those users or items with few interactions. This is a well-known problem known as *cold start*, and it is inherent to the task of a real recommender system, where there will always exist recently created items or users. Nonetheless, when those cases are ignored, it should be made crystal clear and explicitly stated like that in the paper, otherwise the validity of this comparison would not be fair. At the same time, it is interesting to note that cold-start tailored evaluation methodologies have been proposed [Kluver and Konstan (2014)], aiming to provide unbiased measurements when evaluating users with a limited number of training and test groundtruth information. Evaluation in such a scarce situation is not easy, and hence, more studies should be devoted to properly understand any deficiency that may arise in this context.

Additionally, multi-criteria recommendation poses the problem of how to combine various metrics into something that can be optimized [Adomavicius and Kwon (2015)]. Even though there is a significant body of work on multi-criteria RS optimization, there is no clear guideline on how to perform this type of evaluation [Jambor and Wang (2010); Ge *et al.* (2010); Said *et al.* (2013b)]. State of the art methods in multi-criteria evaluation require a trade-off between the different metrics to be evaluated, this can then be applied to offline evaluation. Online evaluation using several criteria requires elaborate qualitative analyses of long term results of recommendation approaches.

Finally, in very sparse datasets such as those used for music recommendation or in location-based RS, it is possible to evaluate by surrogates, which means that a recommendation is assessed as correct if some attribute of the item matches the target item, instead of the actual item, due to the task being extremely difficult and to find the correct item in large catalogs. As stated in [Schedl *et al.* (2015)], evaluation in music RS has been carried out for a long time using genre as proxy and modeling a genre prediction task. Similarly, in location-aware recommendation, the item category has been used as surrogate, assuming that a recommendation would be received by the user in the same way if a museum was recommended, even though that was not the actual museum she visited according to the data [Li *et al.* (2012); Kumar *et al.* (2017)].

These examples evidence that other, more complex environments and evaluation situations can, and should, be defined when evaluating recommendation systems. Either new evaluation metrics, experimental methodologies, or data splitting techniques could be explored to bring closer the RS evaluation to the actual tasks that want to be modeled.

Additional issues related to offline evaluation focus on aspects of the underlying data which is used for both training and evaluation of RSs. The concept known as the *magic barrier* of RS [Herlocker *et al.* (2004)] concerns the fact that user interactions are not concise, i.e., they contain noise and other irregularities which make the data not fully trustworthy. The effect of this is that when optimizing towards a certain metric, there is an upper level, a threshold beyond which optimization is useless [Said *et al.* (2012)]. This threshold is unknown; at best, it can be estimated assuming there are enough interactions provided by each user [Said and Bellogín (2018)].

9.5.2 *Statistical biases on evaluation*

As mentioned in Sec. 9.1.2, while seeking to evaluate recommendation rankings – which largely determine the effective accuracy in matching user needs – rather than predicted rating values, IR metrics have started to be applied for the evaluation of recommender systems. However, as analyzed and described in [Bellogín *et al.* (2017)], two statistical biases arise when adapting this type of measures to RS: popularity bias and sparsity bias. These biases considerably distort the empirical measurements, hindering the interpretation and comparison of results across experiments. Other biases and solutions are surveyed in Chap. **??**.

First, the sparsity bias is related to the fact that the amount of unknown relevance is pervasive in recommendation settings, to a point where some assumptions of the IR methodology may not hold, and the gap between measured and real metric values becomes so significant that a metric absolute magnitude may just lose any meaning. Still, such measurements may support comparative assessments between systems, as far as the bias is system-independent. In this way, depending on the evaluation methodology used, it can be proved whether this bias has an effect on the results. Specifically, the RelPlusN candidate generation strategy inherently controls by design the sparsity bias, by setting the number of sampled non-relevant candidate items; however, other strategies such as TrainItems observe drastic changes in the magnitude of the evaluation metrics, even though the comparison between recommenders is not distorted by test sparsity, meaning that the ranking of algorithms remains stable for different sparsity levels.

On the other hand, the distribution of relevance in RS displays massive popularity skewness patterns that are not found with any comparable strength in IR datasets, which is, thus, not modeled by IR metrics. This phenomenon has a very strong effect not only on metric values, but more importantly on how systems compare to each other, as observed by many authors in recent years [Jannach *et al.* (2015b)]. As presented in [Bellogín *et al.* (2017)], the precision of popularity-based recommendation is heavily determined by the skewness of the distribution; it benefits from steep distributions, and degrades to slightly below random when popularity is uniform. Considering that most of the well-performing personalized algorithms have some kind of popularity bias [Jannach *et al.* (2015b)], this issue also affects those methods. Furthermore, it was observed in [Bellogín *et al.*

(2017)] that any candidate generation strategy that does not distort the inherent item distribution will be affected by this bias.

In this context, the authors from [Bellogín *et al.* (2017)] propose two experimental design approaches that effectively neutralize the popularity bias to a large extent – recall that the sparsity bias can be avoided by simply using the RelPlusN strategy. The first approach consists in partitioning the set of items into m popularity percentiles, breaking down the computation of accuracy by such percentiles, and averaging the m obtained values. By doing so, in a common long-tailed popularity distribution, the margin for the popularity bias is considerably reduced, since the popularity recommender is forced to recommend as many unpopular as popular items, thus leveling the statistical advantage to a significant extent. The second approach, on the other hand, creates data splits where all items have the same amount of test ratings; the assumption here is that the items with a high number of training ratings will no longer have a statistical advantage by having more positive test ratings. According to the experimental results presented in that work, these two approaches are able to reduce the performance of popularity-based algorithms to a random level; moreover, they effectively discriminate better between pure popularity recommenders and algorithms that use popularity as a signal.

9.5.3 *Reproducibility issues on evaluation*

Evaluating recommender systems is not an easy task. In all fairness, it is not *a* task, it is a set of several interconnected and standalone tasks that, when viewed together, should result in one (or several) measure(s) which then state the quality of the recommender. The challenge becomes, for instance, what metrics to use when evaluating, or how many metrics to use.

The results discussed in Sec. 9.4 highlight the differences in recommendation accuracy between implementations of the same algorithms on different frameworks, distinct levels of accuracy and variations of evaluation results on the same dataset and in the same framework when employing various evaluation strategies. It is important to note, thus, that inter-framework comparisons of recommendation quality can potentially point to incorrect results and conclusions, unless performed with great caution and in a controlled, framework independent, environment.

Furthermore, there are several aspects when designing and implementing a recommender system that may affect its final results and hinder a potential replication of the reported settings and insights, from using different model parameters to how specific parts of the system are implemented. For instance, the data splitting strategy may have a deep impact on the final results being reported, making this an important aspect to take into account when reporting details about an experiment. Moreover, there are several aspects in recommendation algorithms that are not standard in the community which, eventually, turn out to be implementation-dependent. For example, the term kNN (to specify collaborative filtering algorithms based on k-nearest neighbors) is usually linked to the user-based (UB) variation of these methods, even though item-based (IB) algorithms are also available and, in some situations, work better than user-based [Sarwar *et al.* (2001)]. Furthermore, there exist several different formulations for the user-based kNN algorithm, none of which is regarded as the "standard" one.

Additionally, there are also different alternatives in the literature when computing neighbors. Although the most typical one consists of taking the top-k closest users to the target user, there are other techniques based on thresholds that should be properly reported and specified when used in experiments because of their unpopularity [Ning *et al.* (2015)]. Besides that, a very important detail that is usually not reported is when the neighbors are actually computed: at training or test (i.e., evaluation) time.

Finally, it is not difficult to find implementations where a capping is applied after the score is predicted – probably inherited by the rating prediction task, the most popular recommendation task for a long time – so such score is bounded by the rating range. However, this may lead to further problems when producing a ranking, since ties are more likely to occur and, hence, tie-breaking strategies have to be implemented and reported, something we have seldom found explicitly in the analyzed frameworks and public implementations. At the same time, it is very important to report what happens when a recommender cannot predict a score. This decision has an impact on the coverage of the system – if a baseline recommender is used instead, every algorithm will always have complete coverage –, but also on how performance metrics are evaluated in those cases.

In summary, the issues of replication – obtaining the exact same results in the same setting – and reproducibility – obtaining comparable results using a different setting – are very difficult challenges at the moment. They force researchers to reimplement the baseline algorithm they want to compare their approach against, or to pay extra attention to every algorithmic and evaluation detail, ignoring if the observed discrepancies with respect to what already was published come from omitted details from the original papers (parameters, methodologies, protocols, etc.) or a wrong interpretation of any of these intermediate steps.

9.6 Future directions

The empirical evaluation of RS is acknowledged to be an open problem in the field, with open issues yet to be addressed [Gunawardana and Shani (2015)]. Many experimental approaches and metrics have been developed over the years, which the community is well acquainted with, but key aspects and details in the design and application of available methodologies are open to configuration and interpretation, where even apparently subtle details may create a considerable difference. This results in a significant divergence in experimental practice, hindering the comparison and proper assessment of contributions and advances to the field.

The discussion and definition of the basic elements of the experimental conditions (and their requirements) are critical to support continuous innovation in any discipline. The offline evaluation of RS requires an implementation of the algorithm or technique to be evaluated, a set of quality measures for comparative evaluation, and an experimental protocol establishing how to handle the data and compute metrics in detail. Online evaluation similarly requires an algorithm implementation and a population of users to survey (by means of an A/B test, for instance). Here again, perhaps even more importantly than in offline evaluation, an experimental protocol needs to be established and adhered to.

In order to seek reproducibility and replication, several strategies can be considered, such as source code sharing, standardization of agreed evaluation metrics and protocols, or releasing public experimental design software, all of which have difficulties of their own. Furthermore, for online evaluation, an extensive analysis of the population of test users should be provided. While the problem of reproducibility and replication has been recognized in the community, the need for a solution remains largely unmet.

Another open problem when evaluating RS is the relation between online and offline experiments. Several authors have explored this issue in some domains but no conclusive results have been obtained [Garcin *et al.* (2014); Beel *et al.* (2013); de Souza Pereira Moreira *et al.* (2015)]. The main question is how to align the offline evaluation to the online (usually, with A/B tests) results. Some possibilities include designing a good evaluation methodology or using a sensible evaluation metric to the problem at hand. An interesting output that could be produced by a better understanding of this issue is that offline evaluation would predict which methods, parameters, or configurations will work better when integrated and tested in an online evaluation. Recent approaches apply counterfactual reasoning to compute offline estimators of business metrics, by computing the expected reward of a new method based on logs collected on a technique running in production [Gilotte *et al.* (2018); Joachims and Swaminathan (2016)]; in this way, we could answer *what if?* questions for a range of algorithms not tested by the users with some level of certainty.

Additionally, there should be an increased effort in making metrics meaningful, especially tailored to specific problems in RS, such as time-aware recommendation, cross-domain, cold-start users and items, repeated consumption, and so on. Several algorithms exist for temporal contexts, however, experimental evaluation methodologies have not been analyzed until recently [Campos *et al.* (2014)], where evaluation dimensions such as novelty and diversity remained largely unexplored, even though there exist obvious relations between these concepts and time-aware recommendations [Lathia *et al.* (2010)]. Similarly, there exist several approaches aiming to exploit cross-domain patterns for recommendation [Cantador *et al.* (2015)], however the evaluation of these systems has been limited to analyze the sensitivity to the amount of user or item overlap, target domain density, and user profile size, neglecting other properties such as the dependence on the groundtruth sparsity levels on either the source or the target domains, or even defining new metrics specifically tailored to this problem, e.g., where the amount of knowledge that is transferred from the source domain is considered inside the metric.

Bibliography

Abel, F. (2015). We know where you should work next summer: Job recommendations, in *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015* (ACM), p. 230.

Adomavicius, G. and Kwon, Y. (2015). Multi-criteria recommender systems, in *Recommender Systems Handbook* (Springer), pp. 847–880.

Adomavicius, G., Tuzhilin, A., Berkovsky, S., De Luca, E. W., and Said, A. (2010). Context-awareness in recommender systems: research workshop and movie recommendation challenge, in *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010* (ACM), pp. 385–386.

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (2011). *Modern Information Retrieval - the concepts and technology behind search, Second edition* (Pearson Education Ltd., Harlow, England), ISBN 978-0-321-41691-9.

Basu, C., Hirsh, H., and Cohen, W. W. (1998). Recommendation as classification: Using social and content-based information in recommendation, in *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA.* (AAAI Press / The MIT Press), pp. 714–720.

Beel, J., Genzmehr, M., Langer, S., Nürnberger, A., and Gipp, B. (2013). A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation, in *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, RepSys 2013, Hong Kong, China, October 12, 2013* (ACM), pp. 7–14.

Bellogín, A., Cantador, I., Díez, F., Castells, P., and Chavarriaga, E. (2013). An empirical comparison of social, collaborative filtering, and hybrid recommenders, *ACM TIST* **4**, 1, p. 14.

Bellogín, A., Castells, P., and Cantador, I. (2011). Precision-oriented evaluation of recommender systems: an algorithmic comparison, in *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago,* *IL, USA, October 23-27, 2011* (ACM), pp. 333–336.

Bellogín, A., Castells, P., and Cantador, I. (2017). Statistical biases in information retrieval metrics for recommender systems, *Information Retrieval Journal* .

Berkovsky, S., Freyne, J., and Coombe, M. (2012). Physical activity motivating games: Be active and get your own reward, *ACM Trans. Comput.-Hum. Interact.* **19**, 4, pp. 32:1–32:41.

Bistaffa, F., Filippo, A., Chalkiadakis, G., and Ramchurn, S. D. (2015). Recommending fair payments for large-scale social ridesharing, in *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015* (ACM), pp. 139–146.

Bollen, D. G. F. M., Knijnenburg, B. P., Willemsen, M. C., and Graus, M. P. (2010). Understanding choice overload in recommender systems, in *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010* (ACM), pp. 63–70.

Bouckaert, R. R. (2003). Choosing between two learning algorithms based on calibrated tests, in T. Fawcett and N. Mishra (eds.), *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA* (AAAI Press), pp. 51–58.

Breese, J. S., Heckerman, D., and Kadie, C. M. (1998). Empirical analysis of predictive algorithms for collaborative filtering, in *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998* (Morgan Kaufmann), pp. 43–52.

Burke, R. D. (2004). Hybrid recommender systems with case-based components, in *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004, Madrid, Spain, August 30 - September 2, 2004, Proceedings*, *Lecture Notes in Computer Science*, Vol. 3155 (Springer), pp. 91–105.

Campos, P. G., Díez, F., and Cantador, I. (2014). Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols, *User Model. User-Adapt. Interact.* **24**, 1-2, pp. 67–119.

Cantador, I., Fernández-Tobías, I., Berkovsky, S., and Cremonesi, P. (2015). Cross-domain recommender systems, in *Recommender Systems Handbook* (Springer), pp. 919–959.

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

Celma, Ò. and Herrera, P. (2008). A new approach to evaluating novel recommendations, in *Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys 2008, Lausanne, Switzerland, October 23-25, 2008* (ACM), pp. 179–186.

Cremonesi, P., Garzotto, F., Negro, S., Papadopoulos, A. V., and Turrin, R. (2011). Comparative evaluation of recommender system quality, in *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Extended Abstracts Volume, Vancouver, BC, Canada, May 7-12, 2011* (ACM), pp. 1927–1932.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks, in *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010* (ACM), pp. 39–46.

de Souza Pereira Moreira, G., de Souza, G. A., and da Cunha, A. M. (2015). Comparing offline and online recommender system evaluations on long-tail distributions, in *Poster Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16, 2015.*, *CEUR Workshop Proceedings*, Vol. 1441 (CEUR-WS.org).

Deshpande, M. and Karypis, G. (2004). Item-based top-N recommendation algorithms, *ACM Trans. Inf. Syst.* **22**, 1, pp. 143–177.

Elahi, M., Ge, M., Ricci, F., Fernández-Tobías, I., Berkovsky, S., and Massimo, D. (2015). Interaction design in a mobile food recommender system, in *Proceedings of the Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, IntRS 2015, co-located with ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 19, 2015.*, *CEUR Workshop Proceedings*, Vol. 1438 (CEUR-WS.org), pp. 49–52.

Elahi, M., Ge, M., Ricci, F., Massimo, D., and Berkovsky, S. (2014). Interactive food recommendation for groups, in *Poster Proceedings of the 8th ACM Conference on Recommender Systems, RecSys 2014, Foster City, Silicon Valley, CA, USA, October 6-10, 2014*, *CEUR Workshop Proceedings*, Vol. 1247 (CEUR-WS.org).

Elsweiler, D., Harvey, M., Ludwig, B., and Said, A. (2015). Bringing the "healthy" into food recommenders, in *Proceedings of the 2nd International Workshop on Decision Making and Recommender Systems, Bolzano, Italy, October 22-23, 2015.*, *CEUR Workshop Proceedings*, Vol. 1533 (CEUR-WS.org), pp. 33–36.

Garcin, F., Faltings, B., Donatsch, O., Alazzawi, A., Bruttin, C., and Huber, A. (2014). Offline and online evaluation of news recommender systems at swissinfo.ch, in *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014* (ACM), pp. 169–176.

Ge, M., Delgado-Battenfeld, C., and Jannach, D. (2010). Beyond accuracy: evaluating recommender systems by coverage and serendipity, in *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010* (ACM), pp. 257–260.

Gilotte, A., Calauzènes, C., Nedelec, T., Abraham, A., and Dollé, S. (2018). Offline A/B testing for recommender systems, in Y. Chang, C. Zhai, Y. Liu, and Y. Maarek (eds.), *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018* (ACM), pp. 198–206.

Goldberg, K. Y., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm, *Inf. Retr.* **4**, 2, pp. 133–151.

Gunawardana, A. and Shani, G. (2015). Evaluating recommender systems, in *Recommender Systems Handbook* (Springer), pp. 265–308.

Guy, I. (2015). Social recommender systems, in *Recommender Systems Handbook*

(Springer), pp. 511–543.

Harper, F. M. and Konstan, J. A. (2016). The movielens datasets: History and context, *TiiS* **5**, 4, pp. 19:1–19:19.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. (2004). Evaluating collaborative filtering recommender systems, *ACM Trans. Inf. Syst.* **22**, 1, pp. 5–53.

Jambor, T. and Wang, J. (2010). Optimizing multiple objectives in collaborative filtering, in *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010* (ACM), pp. 55–62.

Jannach, D., Lerche, L., and Jugovac, M. (2015a). Adaptation and evaluation of recommendations for short-term shopping goals, in *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015* (ACM), pp. 211–218.

Jannach, D., Lerche, L., Kamehkhosh, I., and Jugovac, M. (2015b). What recommenders recommend: an analysis of recommendation biases and possible countermeasures, *User Model. User-Adapt. Interact.* **25**, 5, pp. 427–491.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques, *ACM Trans. Inf. Syst.* **20**, 4, pp. 422–446.

Jäschke, R., Marinho, L. B., Hotho, A., Schmidt-Thieme, L., and Stumme, G. (2007). Tag recommendations in folksonomies, in *Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007, Proceedings*, *Lecture Notes in Computer Science*, Vol. 4702 (Springer), pp. 506–514.

Joachims, T. and Swaminathan, A. (2016). Counterfactual evaluation and learning for search, recommendation and ad placement, in R. Perego, F. Sebastiani, J. A. Aslam, I. Ruthven, and J. Zobel (eds.), *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016* (ACM), pp. 1199–1201.

Kluver, D. and Konstan, J. A. (2014). Evaluating recommender behavior for new users, in *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014* (ACM), pp. 121–128.

Knijnenburg, B. P. and Willemsen, M. C. (2015). *Evaluating Recommender Systems with User Experiments* (Springer US, Boston, MA), ISBN 978-1-4899-7637-6, pp. 309–352.

Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. (2009). Controlled experiments on the web: survey and practical guide, *Data Min. Knowl. Discov.* **18**, 1, pp. 140–181.

Konstan, J. A. and Adomavicius, G. (2013). Toward identification and adoption of best practices in algorithmic recommender systems research, in *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, RepSys 2013, Hong Kong, China, October 12, 2013* (ACM), pp. 23–28.

Koren, Y. and Bell, R. M. (2015). Advances in collaborative filtering, in *Recommender Systems Handbook* (Springer), pp. 77–118.

Kosir, A., Odic, A., and Tkalcic, M. (2013). How to improve the statistical power of the 10-fold cross validation scheme in recommender systems, in A. Bellogín, P. Castells, A. Said, and D. Tikk (eds.), *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, RepSys 2013, Hong Kong, China, October 12, 2013* (ACM), pp. 3–6.

Kumar, G., Jerbi, H., and O'Mahony, M. P. (2017). Towards the recommendation of personalised activity sequences in the tourism domain, in *Proceedings of the 2nd Workshop on Recommenders in Tourism co-located with 11th ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27, 2017.*, *CEUR Workshop Proceedings*, Vol. 1906 (CEUR-WS.org), pp. 26–30.

Lathia, N., Hailes, S., Capra, L., and Amatriain, X. (2010). Temporal diversity in recommender systems, in *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010* (ACM), pp. 210–217.

Li, W., Eickhoff, C., and de Vries, A. P. (2012). Want a coffee?: predicting users' trails, in *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012* (ACM), pp. 1171–1172.

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering, *IEEE Internet Computing* **7**, 1, pp. 76–80.

Luo, L., Li, B., Berkovsky, S., Koprinska, I., and Chen, F. (2016). Who will be affected by supermarket health programs? tracking customer behavior changes via preference modeling, in *Advances in Knowledge Discovery and Data Mining - 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part I*, *Lecture Notes in Computer Science*, Vol. 9651 (Springer), pp. 527–539.

McLaughlin, M. R. and Herlocker, J. L. (2004). A collaborative filtering algorithm and evaluation metric that accurately model the user experience, in *SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 25-29, 2004* (ACM), pp. 329–336.

McNee, S. M., Riedl, J., and Konstan, J. A. (2006). Being accurate is not enough: how accuracy metrics have hurt recommender systems, in *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006* (ACM), pp. 1097–1101.

Mesas, R. M. and Bellogín, A. (2017). Evaluating decision-aware recommender systems, in *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017* (ACM), pp. 74–78.

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

Pu, P., Chen, L., and Hu, R. (2012). Evaluating recommender systems from the user's perspective: survey of the state of the art, *User Model. User-Adapt. Interact.* **22**, 4-5, pp. 317–355.

Said, A. (2016). A short history of the recsys challenge, *AI Magazine* **37**, 4, pp. 102–104.

Said, A. and Bellogín, A. (2014). Comparative recommender system evaluation: benchmarking recommendation frameworks, in *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014* (ACM), pp. 129–136.

Said, A. and Bellogín, A. (2018). Coherence and inconsistencies in rating behavior: estimating the magic barrier of recommender systems, *User Modeling and User-Adapted Interaction* .

Said, A., Bellogín, A., Lin, J. J., and de Vries, A. P. (2014a). Do recommendations matter?: news recommendation in real life, in *Computer Supported Cooperative Work, CSCW '14, Baltimore, MD, USA, February 15-19, 2014, Companion Volume* (ACM), pp. 237–240.

Said, A., Fields, B., Jain, B. J., and Albayrak, S. (2013a). User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm, in *Computer Supported Cooperative Work, CSCW 2013, San Antonio, TX, USA, February 23-27, 2013* (ACM), pp. 1399–1408.

Said, A., Jain, B. J., and Albayrak, S. (2013b). A 3d approach to recommender system evaluation, in *Computer Supported Cooperative Work, CSCW 2013, San Antonio, TX, USA, February 23-27, 2013, Companion Volume* (ACM), pp. 263–266.

Said, A., Jain, B. J., Narr, S., and Plumbaum, T. (2012). Users and noise: The magic barrier of recommender systems, in *User Modeling, Adaptation, and Personalization - 20th International Conference, UMAP 2012, Montreal, Canada, July 16-20, 2012. Proceedings*, *Lecture Notes in Computer Science*, Vol. 7379 (Springer), pp. 237–248.

Said, A., Tikk, D., and Cremonesi, P. (2014b). Benchmarking - A methodology for ensuring the relative quality of recommendation systems in software engineering, in *Recommendation Systems in Software Engineering* (Springer), pp. 275–300.

Sakai, T. (2014). Statistical reform in information retrieval? *SIGIR Forum* **48**, 1, pp. 3–12.

Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms, in *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001* (ACM), pp. 285–295.

Schedl, M., Knees, P., McFee, B., Bogdanov, D., and Kaminskas, M. (2015). Music recommender systems, in *Recommender Systems Handbook* (Springer), pp. 453–492.

Smyth, B. and McClave, P. (2001). Similarity vs. diversity, in *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30 - August 2, 2001, Proceedings*, *Lecture Notes in Computer Science*, Vol. 2080

(Springer), pp. 347–361.

Tomlinson, S. (2012). Measuring robustness with first relevant score in the TREC 2012 microblog track, in *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012*, Vol. Special Publication 500-298 (National Institute of Standards and Technology (NIST)).

Zhao, X., Zhang, W., and Wang, J. (2015). Risk-hedged venture capital investment recommendation, in *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015* (ACM), pp. 75–82.