A Closer-to-Reality Model for Comparing Relevant Dimensions of Recommender Systems, with Application to Novelty

François Fouss * and Elora Fernandes

Louvan Research Institute in Management and Organizations (LouRIM), UCLouvain, Chaussée de Binche 151, 7000 Mons, Belgium; elora.fernandes@uclouvain.be
* Correspondence: francois.fouss@uclouvain.be

Abstract: Providing fair and convenient comparisons between recommendation algorithms—where algorithms could focus on a traditional dimension (accuracy) and/or less traditional ones (e.g., novelty, diversity, serendipity, etc.)—is a key challenge in the recent developments of recommender systems. This paper focuses on novelty and presents a new, closer-to-reality model for evaluating the quality of a recommendation algorithm by reducing the popularity bias inherent in traditional training/test set evaluation frameworks, which are biased by the dominance of popular items and their inherent features. In the suggested model, each interaction has a probability of being included in the test set that randomly depends on a specific feature related to the focused dimension (novelty in this work). The goal of this paper is to reconcile, in terms of evaluation (and therefore comparison), the accuracy and novelty dimensions of recommendation algorithms, leading to a more realistic comparison of their performance. The results obtained from two well-known datasets show the evolution of the behavior of state-of-the-art ranking algorithms when novelty is progressively, and fairly, given more importance in the evaluation procedure, and could lead to potential changes in the decision processes of organizations involving recommender systems.

Keywords: novelty; popularity bias; data splitting; algorithms comparison

1. Introduction

The goal of recommender systems is to be as good as possible when suggesting items to people. However, you can be sure that if you ask each of us the question “What is a good recommendation?” you will get as many different answers as there are respondents—for example, the one that makes the user happy, the one that directs the user to an interesting, novel, surprising, or new item, or even the one that maximizes the sales margin of the seller, etc. From the perspective of the literature, however (see, among others, [1–3]), the evaluation of the quality of a recommender system has been performed mainly by accuracy metrics for many years, although some authors (see, among others, [4–6]) have introduced a broad set of properties (other than accuracy) that are relevant to system success. Those accuracy metrics evaluate the relevance of the items that have been proposed mainly by accuracy metrics for many years, although some authors (see, among others, [4–6]) have introduced a broad set of properties (other than accuracy) that are relevant to system success. Evaluations methods such as Root Mean Standard Error (RMSE) or Mean Absolute Error (MAE) measure the accuracy for a user–item pair (i.e., rating prediction) when other metrics such as Prediction, Recall, F1, or Exponential Decay measure the accuracy of a user–list pair (i.e., ranking) [7]. This work focuses on systems recommending a ranked list to each user and therefore not on rating predictions.

1.1. Evaluation Metrics

Obviously, recommender systems should continue to pay attention to accuracy, as the focus on accuracy helps to reduce prediction errors. In many areas, for example, it seems crucial to avoid bad recommendations (i.e., in terms of accuracy) as, among other considerations, this could influence the perceived quality of the system [8].
therefore focus on avoiding inaccurate recommendations, which are considered risky and dangerous (even if this means avoiding new or unexpected items [1,9,10]). From the user’s perspective, the quality of a recommendation is, however, progressively shifting from a strictly accuracy-based dimension to other dimensions not exclusively focused on accuracy (such as, e.g., novelty, diversity, serendipity, etc.), leading to a growing number of papers showing that accuracy is no longer sufficient for evaluating such a system “goodness” [11].

However, in the literature (see, for example, [1–3,12–15]), the evaluation and the comparison of the quality of a recommender system is still heavily dominated by accuracy. Indeed, due to the way they are optimized (i.e., in maximizing/minimizing accuracy metrics, in a standard crossvalidation framework), recommender systems are mainly penalized in accuracy evaluation when they suggest items that are relevant but whose relevance is related to these non-accuracy-focused dimensions (even though users increasingly give credit to these dimensions), leading to the exclusive use of accuracy measures as evaluation metrics, or by supplementing (rather than integrating) accuracy-optimized metrics with non-accuracy-based metrics.

1.2. Evaluation Process

In his recent work [16], Riley identified three pitfalls to avoid in learning processes, such as in training a recommender system, one of them being splitting data inappropriately. When building models, practitioners typically divide data into training and test sets. The training set is used to train the model, whose performance is evaluated on the test set. Researchers usually split the data randomly. However, in real life, data are rarely random.

When evaluating the accuracy of a recommender system [17,18], a random split of the set of user–item interactions (training interactions vs. test interactions) is traditionally performed, either by crossvalidation or by bootstrapping. Furthermore, in a random split, the number of test ratings for an item is correlated with its number of training ratings (see [19]). Popular items therefore predominate in a regular test set. The degree of skewness in the distribution of ratings greatly influences the measured performance of a system.

Thus, accuracy metrics are heavily biased toward popular items [20] and their inherent dominant features—i.e., if most of us like science-fiction movies, science-fiction movies will dominate the (accuracy-based) recommendations for all the users as they dominate the test set. The authors showed that, when analyzing the performance of recommender algorithms on top-n recommendation tasks, the few most popular items can skew the top-n performance, and that the test set should be chosen carefully in order not to bias accuracy metrics toward nonpersonalized solutions. This is further reinforced by the work of Bellogín [21], who argues that popularity bias significantly distorts the empirical measurements, hindering the interpretation and comparison of results across experiments.

One of the current challenges is therefore to find a way to evaluate recommender systems that reflects the true quality of the system—quality in terms of accuracy but also in terms of new dimensions that consider the “nonobviousness” (term introduced in [22]) of the recommendation, such as novelty, diversity, etc.

This work, which is based on these two statements (i.e., the importance of splitting the data appropriately and the reduction of the evaluation bias toward nonpersonalized solutions), presents a new model for evaluating the quality of a recommender system by working on the popularity bias inherent in traditional training/test set frameworks, where test sets are built randomly and thus biased by dominant popular items. To the best of our knowledge, there is no work comparing recommender systems without using traditional training/test set frameworks that are biased by popularity. In our model, each item has a probability of being included into the test set that depends on a probability distribution based on a specific feature related to the targeted dimension (i.e., novelty in this work). Our work aims to partially address the need to validate non-accuracy-focused algorithms in a fair way (i.e., compared to accuracy-focused algorithms). In addition, our model, which is inspired by validation processes where implicit feedback is available (through direct contact with users) postulates that, if the aim is to evaluate whether a
recommender system is accurate in suggesting novel items, this should not be performed (as is usually the case in the literature, see, for example, [1–3,14,15]) by providing popular items as test examples, but rather more novel items (thus showing the actual ability of the system to be accurate in recommending novel items), which is precisely the goal of our approach. Note therefore that our model does not claim to deal with biases inherent in existing datasets (such as, for example, the two datasets used in this work—see Section 4.1), or biases inherent in existing algorithms (for such biases, see for example the recent work on fairness for recommender systems [23–25]), but rather to deal with popularity bias related to the evaluation processes traditionally applied in the field of recommender systems (i.e., random crossvalidation or bootstrapping), which leads to the fact that the joint comparison of accuracy and another dimension (such as, for example, novelty) by traditional processes is not fair.

The goal of this work is to reconcile, in terms of evaluation (and thus comparison), accuracy-based and novelty-based algorithms, by comparing their accuracy and novelty performance in an integrated model. Our research questions could therefore be formulated as follows: “How does the performance of state-of-the-art, optimized on an accuracy basis, recommendation algorithms evolve when their evaluation process does not include an artificial bias toward highly popular items and is therefore closer to real processes?” and “How does the performance-based ranking between recommendation algorithms evolve (and hence the choice of which recommendation algorithm to choose in real applications) when their evaluation process does not include an artificial bias toward very popular items and is therefore closer to real processes?”

The contributions of this paper are threefold

• To propose a new model closer to reality, integrating accuracy and novelty, to evaluate the performance of recommendation algorithms;
• To propose a new perspective for comparing recommendation algorithms;
• To provide guidance on which algorithms to use in real-world recommendation applications involving both accuracy and novelty.

We first provide an overview of related research work in Section 2. Section 3 describes the proposed model. Section 4 presents the experimental methodology used for our tests and the generic metric used to quantify novelty. Section 5 introduces different, well-known algorithms used for recommendation. The results are shown in Section 6 and discussed in Section 7, while Section 8 concludes the paper.

2. Related Work

In [26], the authors discuss the next generation of recommender systems, which will be novelty driven. They argue that recommender systems have drawbacks that could seriously reduce our open-mindedness and ability to experience diversity, claiming that it is possible to overcome the limitations of current recommender systems, by obtaining inspiration from the way in which people seek for novelties and give values to new experiences. Nandal et al. go even further in [27], writing that less novelty is the main reason for users’ increased frustration when searching for something new.

2.1. Defining Novelty

Novelty is commonly perceived as the quality of being different, new, and interesting [28] (see [29,30] for a first discussion on novelty, in information retrieval).

Firstly, an item could be considered as novel if it is different from what the user has seen before (i.e., the degree to which an item is unusual to the users’ usual tastes [31]). This aspect of novelty is usually quantified by measuring a distance between the “novel” item and the set of previously consumed items.

Secondly, an item could be considered as novel if it is unknown [22]. Vargas and Castells [32] define novelty as something new, i.e., that has not been experienced before. Note that some authors put “unknown” and “unexpectedness” in parallel [33] as the importance of nonobviousness in novel items has been emphasized [22].
Finally, other authors emphasize the interest for items in their work on novelty (see, e.g., [30], where the user’s interest for an item is integrated into the concept of novelty, assuming that the quality of an item is only noticed when it is relevant).

2.2. Quantifying Novelty

Without any kind of user direct feedback (e.g., through interviews or surveys), it is hardly possible to verify whether an item is novel for a user. With this in mind, we notice two main perspectives for measuring novelty when no user feedback is available. Metrics can be defined either locally for each user or globally for all users.

On the one hand, many methods are based on the assumption that novelty is a user-dependent concept. Indeed, an item could be novel for one user and not for another, or furthermore, users are not equal in terms of their propensity to like novel items. These methods involve tailoring the novelty recommendation strategy for each user (see, e.g., [34,35]) and usually depend on metrics based on a distance to the subset of items with which the user has previously interacted.

On the other hand, other methods, though, suggest considering novelty as a general principle—applied in the same way for all the users (see, e.g., [36–38], etc.), thus considering the community as a whole, and usually depending on metrics based on the global popularity of the item (therefore opposing novelty and popularity, such as in [39]). The underlying assumption is that a user is more likely to know about a popular item than an unpopular item [20,32,40]. Novelty in this case is inversely proportional to the global popularity of an item [36].

2.3. Alternative Metrics

For a long time, any recommender system—regardless of its purpose, was evaluated exclusively on the basis of accuracy metrics, which is now being questioned.

Del Olmo and Gaudioso [12] highlight the challenge of both evaluating and comparing systems, as metrics should face a range of broad questions to be selected. Selecting the appropriate metric is critical [11]. In recent years, many authors have proposed methods that focus on dimensions other than accuracy [41]. Most of these works either compare metrics for evaluation [32,36,42]—e.g., accuracy on the one hand and diversity, novelty, etc., on the other hand—or use multiobjective optimizations [43,44], or even combine metrics into new metrics [2].

Maksai et al. [41] combine accuracy, diversity, coverage, and serendipity metrics to create a new performance model. To correct a potential popularity bias in the available data—that is, users are more likely to provide feedback on mainstream (popular) items than on niche (unpopular) items, leading to the fact that observed feedback does not reflect users’ actual interests and is biased toward popular items [45]—Steck [46] merged an accuracy metric and the item popularity into a single metric (the popularity-stratified recall) by introducing into the usual recall measure a weight proportional to the inverse probability of an item obtaining feedback (assuming that users are more likely to provide feedback on popular items than on unpopular items).

Shi et al. [47] propose to cancel the effect of the most popular items so that they do not contribute to any of the evaluation metrics. Based on the same idea, Shani et al. [7] argue that more attention should be paid to less popular items and suggest adjusting accuracy metrics so that higher scores are given to successful predictions of less popular items.

2.4. Data Splitting

Three steps are generally common to the functioning of recommender systems: gathering valuable information about users and items, determining patterns from historical data, and recommending items to users. To assess the quality of these recommendations, a k-fold crossvalidation or bootstrapping is usually performed, and the results, averaged over the k fold, are traditionally provided. Basically, such a process is a matter of randomly splitting the data into subsets which are then used either for training (usually by using a
larger proportion of the data) or for testing (usually on a smaller proportion of the data) but not for both.

This data splitting strategy is therefore an element of the evaluation process. Common strategies in recommender systems are presented in [48] and include firstly temporal approaches (i.e., considering the timing of events), where a basic approach is to select a point in time to separate training and test data (see, e.g., [49]). If time is not taken into account, sampling methods include using the occurrences for each user as a test, determined randomly as a fixed number or as a percentage of the interactions, which leads to the classical processes of random crossvalidation and bootstrapping.

To address the popularity bias identified in the random construction of training and test sets (see Section 1), Bellogín et al. [19,21] developed two approaches. The first one—a percentile-based approach—consists of dividing the set of available items into several popularity percentiles and then breaking down the accuracy calculation by percentiles. The second one—a uniform test item profiles—forms the data split so that all items have the same number of test ratings. In other words, each item is equally represented in the test set, regardless of its popularity.

In addition, Cremonesi et al. [20] partition the test set into two subsets: the head (popular items) and the long-tail (unpopular items) in order to evaluate the accuracy of recommender systems in suggesting nontrivial items. In [50], a novelty value is used to generate test sets of different levels of difficulty on which to measure the accuracy performance of the system by selecting items in each user’s test set with equal probability from the top \( c \% \) of the most novel items (defined by a novelty metric) in the user’s profile (with \( c \) as parameter).

3. Proposed Model

Traditionally, as detailed in Section 2.4, the available data on user–item interactions are randomly divided, either by crossvalidation or bootstrapping, into a training set (to train the model) and a test set (to evaluate the model) according to a specified rule. In other words, these traditional data-splitting frameworks rely on a uniform probability distribution of picking each user–item interaction as a test example, which leads to an over-representation of popular items (as they are over-represented in user–item interactions).

We propose a process for evaluating recommender systems algorithms that limits this artificial popularity bias so that different algorithms can be compared on a basis closer to reality in terms of both accuracy and novelty. We suggest that, to avoid test sets consisting mainly of popular items, instead of a uniform distribution, a Poisson distribution should be used to distribute the user–item interactions between the training and the test sets.

More precisely, all items in an initial dataset are sorted according to their decreasing popularity (an item rated by many users is considered popular) and divided into 20 equally sized groups of popularity—later corresponding to 20 different probabilities of integrating the test set. Each group is further associated with a value of parameter \( k \) (\( k = 0 \ldots 19 \)), \( k = 0 \) for the group containing the most popular items. Note that other choices (i.e., than 20) could have been made for the number of groups. As detailed in Section 4.1 describing the datasets and showing the impact of group decomposition on the test sets, the choice of 20 groups seems, however, to be a fair choice for such “long-tailed” datasets.

The probability that a user–item interaction integrates the test set depends, on the one hand, on the affiliation to the group of items (\( k \)) and, on the other hand, on the rate parameter \( \lambda \) of the Poisson distribution (we propose three values for parameter \( \lambda \): 2, 4, and 6). Following a Poisson distribution, the probability of observing events of popularity \( k \) in an interval is given by:

\[
P(k) = \frac{\lambda^k e^{-\lambda}}{k!}
\]

The choice of using a Poisson distribution was inspired by the work conducted in the field of information retrieval on “term frequency-inverse document frequency” (tf-idf, see, e.g., [51]), where words appearing in many documents (i.e., very popular words) have a
low idf score, and thus tf-idf, while words appearing in very few documents (i.e., very unpopular words) have a high idf score, and thus a low tf-idf. Tf-idf was adapted to our context by considering the most popular items and the most unpopular items as less discriminant for estimating tastes of users.

The precise consequences of using such a data-splitting process depend on the initial dataset (and the popularity of its items) and are further developed in Section 4.1 on datasets.

4. Experimental Methodology

4.1. Datasets

This work is applied to two distinct datasets: MovieLens and Book-Crossing. MovieLens (ML) is a dataset of real movies from the online recommender system MovieLens (https://movielens.org/, accessed on 26 November 2021). We performed our experiments on the version of the dataset containing 100,000 ratings given by 943 users—users who rated at least 20 movies were selected—to 1682 movies. The user–movie interactions matrix contains around 6.3% of “1” values and 93.7% of “0” values. We also use a sample of the Book-Crossing (BC) dataset [42] containing 2222 books, 1028 persons, and 109,374 ratings (retaining only those people who rated 40 or more books and books that were rated by 20 or more people). The user–book interactions matrix contains around 4.8% of “1” values and 95.2% of “0” values.

Figure 1 shows (for random crossvalidation, random bootstrap, and the three tested values of $\lambda$ for the Poisson distribution) the proportion of interactions, related to items of group $k$, in the test set. As shown in Figure 1, highly popular and highly unpopular items have a lower probability of being picked up in a test set when following a Poisson distribution, compared to, first, random crossvalidation or random bootstrapping and, second, items in the middle part of the curve (i.e., less popular than bestsellers but more popular than unknown items). Moreover, moving from $\lambda = 2$ to $\lambda = 6$ leads to giving less weight to the rather popular (but not the most popular) items and more weight to the less popular (but still frequently appearing) items. Indeed, we see that, for $\lambda = 2$, the most represented popularity groups are $k = 1$ and $k = 2$ with an associated probability of 0.2707. By increasing the value of the parameter $\lambda$, the distribution becomes more symmetrical, flatter, and right-sided. Thus, the maximum of 0.1954 probability is reached for $k = 3$ and $k = 4$ with $\lambda = 4$, and for $k = 5$ and $k = 6$, with $\lambda = 6$, with a probability of 0.1606.

![Figure 1. Proportion of interactions, related to items of group $k$, in the test set (ML and BC).](image)

Moreover, as for each bootstrap/crossvalidation step, the training set is the complementary set of the corresponding test set, if highly popular and highly unpopular items have a lower probability of being picked up in a test set when following a Poisson distribution, this means that the corresponding training set could be biased in the opposite direction.
To avoid such an undesirable effect, each complementary training set was adapted so that its distribution of interactions remains as close as possible to the original distribution of the dataset, by randomly removing over-represented interactions (due to the building process of the test set).

4.2. Evaluation Metrics
4.2.1. Accuracy
This work focuses on algorithms recommending a ranked list to each user (and not on algorithms predicting ratings). The accuracy of these algorithms can be measured using the normalized discounted cumulative gain metric (NDCG—see, e.g., [52]—the gain is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks) or the recall metric—conventionally, the ratio of relevant items that are recommended in the predicted list [19]. Both NDCG and recall scores were computed on a, standard, top 20 ranked list, and averaged on all individuals. Note that, as no difference between the two metrics was found in the results, only NDCG is reported and discussed in the sequel.

4.2.2. Novelty
We refer to the Expected Popularity Complement (EPC) [32]—a metric commonly used when working without user feedback—to measure the novelty of a recommendation list $R$ (composed of items $i$, each ranked in a position $l$) proposed to a user $u$.

$$\text{Score}_{\text{nov}}(R|u) = \text{EPC} = \alpha \sum_{i \in R} \text{nov}(i|\Theta) \times \text{disc}(l) \times \text{rel}(i,l,u)$$

where
- $\alpha$ is a normalizing constant, fixed to $\frac{1}{n^2}$,
- $\text{nov}(i|\Theta)$ is the novelty of item $i$ in a particular context $\Theta$ and can be measured in various ways (see Item Novelty paragraph);
- $\text{disc}(l)$ is the estimation of an item discovery depending on its position $l$ in the recommendation list (see Item Discovery paragraph);
- $\text{rel}(i,l,u)$ is the item relevance meaning the interest of a user $u$ for a specific item $i$ (see Item Relevance paragraph).

In the sequel, these variants of EPC are referred to as $\text{EPC}_{X,Y,Z}$ where $X$ relates to $\text{nov}(i|\Theta)$, $Y$ to $p(\text{seen}|i,\Theta)$, and $Z$ to $\text{disc}(l)$. A summary of all these variants is available in Table 1.

Item Novelty
We refer to item novelty as a Popularity-Based Item Novelty [32] where the novelty of item $i$ depends on the probability that $i$ was not known prior to the recommendation (see Equations (3) and (4) for its logarithmic version, aiming at emphasizing highly novel items). Less popular items—those that have received the attention of only a few users—are more likely to be novel.

$$\text{nov}(i|\Theta) = 1 - p(\text{seen}|i,\Theta)$$

$$\text{nov}(i|\Theta) = -\log_2(p(\text{seen}|i,\Theta))$$

In the sequel, using Equation (3) or (4) is referred, in our nomenclature, respectively, by $X = 1$ (and therefore by $\text{EPC}_{1,Y,Z}$) for Equation (3) and $X = 2$ (and thus by $\text{EPC}_{2,Y,Z}$) for Equation (4).

In their work, Vargas and Castells [32] also suggest that item novelty could be evaluated by the Inverse User Frequency (IUF) (see Equation (5)). IUF was originally introduced by Breese et al. [53] to penalize predicted rating of popular items (and then promote unpopular items):

$$\text{nov}(i|\Theta) = \log \frac{n}{n_i}$$
where \( n_i \) is the number of users who have rated item \( i \) out of the total number of users \( n \).

In the sequel, using Equation (5) is referred, in our nomenclature, by \( X = 3 \) (and thus by \( \text{EPC}_{3,Z} \) (note that, in this case, there is no use of \( p(\text{seen}|i, \Theta) \) (unlike Equation (3) or (4)), and therefore, no parameter was needed for \( Y \)—see next paragraph) for Equation (5).

Furthermore, for Equations (3) and (4), the work now consists in estimating \( p(\text{seen}|i, \Theta) \), related to \( Y \) in our nomenclature. In accordance with the literature, and being constrained by the fact that we could not evaluate whether a recommended item is really unknown to users (no feedback), we propose two ways of estimating \( p(\text{seen}|i, \Theta) \):

1. Based on the number of user–item interactions observed among the interactions [54]:

\[
p(\text{seen}|i, \Theta) = \frac{n_i - \min_{\text{view}}}{\max_{\text{view}} - \min_{\text{view}}} \quad (6)
\]

where \( \min_{\text{view}} \) is the number of ratings for the less-rated item and \( \max_{\text{view}} \) is for the most-rated item, such that \( p(\text{seen}|i, \Theta) \) ranges between 0 and 1. This is referred in the sequel by \( Y = 1 \) and therefore \( \text{EPC}_{X,1,Z} \).

2. Based on the fraction of total users who rated item \( i \) [28]:

\[
p(\text{seen}|i, \Theta) = \frac{n_i}{n} \quad (7)
\]

where \( n_i \) is the number of users who rated item \( i \) out of the total number of \( n \) users. This is referred in the sequel by \( Y = 2 \) and therefore \( \text{EPC}_{X,2,Z} \).

**Item Discovery**

The position of an item in a recommendation list can influence whether a user effectively sees that item when browsing through the recommendation list \( R \). A user is more likely to notice a higher-ranked item than a lower-ranked item.

To estimate the item discovery parameter \( \text{disc}(l) \), related to \( Z \) in our nomenclature, two alternatives are used, one providing equal weights to each item within a list of recommendations, assuming that users pay attention to each of the 20 items of the list \( R \) (i.e., \( \text{disc}(l) = 1 \)) (which is referred in the sequel by \( Z = 1 \) and therefore \( \text{EPC}_{X,Y,1} \)), and the other suggested by Vargas and Castells [32], where \( \text{disc}(l) = 0.85l^{-1} \) (which is referred in the sequel by \( Z = 2 \) and therefore \( \text{EPC}_{X,Y,2} \)).

**Item Relevance**

Relevance is a user-specific notion which is related to the user’s interests [32]. A relevant item for a user \( u \) might not be relevant for another user \( v \) who has different tastes. If an item is liked, useful, appreciated, etc., by a user, it is considered relevant to that particular user.

As our experiment was based on real data but without possible feedback from users, we decided to keep the relevance score at a fixed value, i.e., \( \text{rel}(i, u) = 1 \), for all \( i, u \) and position \( l \) of the item in the list, thus assuming that the top 20 recommended items are equally relevant for the user. This relevance parameter is therefore not included in our \( \text{EPC}_{X,Y,Z} \) nomenclature.
Table 1. EPC variants.

| EPC_{X,Y,Z} | X     | nov(i|Θ) | Y     | p(seen|j, Θ) | Z     | disc(l) |
|-------------|-------|--------|-------|------------|-------|---------|
| EPC_{1.1.1} | 1.1   | 1      | 1     | n|n|n|min_n|max_n|n|n|n|1|1|
| EPC_{1.1.2} | 2.05 | 1 - p(seen|j, Θ) | 2     | n|n|n|min_n|max_n|n|n|n|1|1|
| EPC_{1.2.1} | 2.05 | 1     | 1     | n|n|n|min_n|max_n|n|n|n|1|1|
| EPC_{1.2.2} | 2.05 | 1     | 1     | n|n|n|min_n|max_n|n|n|n|1|1|
| EPC_{2.1.1} | 2.05 | 1     | 1     | n|n|n|min_n|max_n|n|n|n|1|1|
| EPC_{2.1.2} | 2.05 | 1     | 1     | n|n|n|min_n|max_n|n|n|n|1|1|
| EPC_{2.2.1} | 2.05 | 1     | 1     | n|n|n|min_n|max_n|n|n|n|1|1|
| EPC_{2.2.2} | 2.05 | 1     | 1     | n|n|n|min_n|max_n|n|n|n|1|1|
| EPC_{3.1}   | 2.05 | 1     | 1     | n|n|n|min_n|max_n|n|n|n|1|1|
| EPC_{3.2}   | 2.05 | 1     | 1     | n|n|n|min_n|max_n|n|n|n|1|1|

5. Algorithms

This section presents the different algorithms which were implemented in our experiments (see Table 2 for a summary).

Table 2. Algorithms Summary.

| Algorithms                          | Abb. | Parameters | Values            |
|-------------------------------------|------|------------|-------------------|
| Matrix Factorization                | MF   | Latent Factor f | 40                |
| User-Based Collaborative Filtering  | CF   | Number of neighbors k | 20                |
| Latent Class Model                  | LC   | Latent Class z | 20 (ML)—15 (BC)  |
| Inverted Recommendations            | IR   | Number of neighbors k | 20                |
| Reranking                           | RR   | Ranking Threshold T_R | MaxScore/2        |
| Explorimenter                       | XP   | Number of neighbors k | 20                |

5.1. Classic Algorithms

5.1.1. Matrix Factorization (MF)

Matrix factorization reduces the dimensionality of a matrix as it decomposes a n \times m matrix into a product of two lower dimensionality matrices V and W, of sizes n \times f and f \times m, respectively, mapping both users and items to a joint latent factor space of dimension f, so that user–item interactions are modeled as inner products in this space [55].

In our case, n corresponds to the total number of users and m to the total number of items. Each user \( u \) is represented by a 1 \times f row vector \( v_u \), whose elements reflect the user’s interest in each latent factor \( f \), while each item \( i \) is represented by a f \times 1 column vector \( w_i \), whose elements reflect the item’s connection with each latent factor \( f \).

The user’s overall interest in the features of the item \( \hat{r}_{ui} \)—the expected rating for the item by the user—can be computed (see [55]) by

$$\hat{r}_{ui} = v_u w_i \tag{8}$$

As shown in [56], an increase in the number of latent factors usually improves accuracy, especially for sparse datasets with, as a counterpart, an increase in the computational complexity of the matrix factorization, which is proportional to f. Therefore, as suggested in [57]—where authors experiment from \( f = 20 \) to \( f = 100 \)—a balance between accuracy and efficiency has to be considered, and we set the number of latent factors to 40.

5.1.2. User-Based Collaborative Filtering (CF)

CF approaches are based on the assumption that people who agreed in the past tend to agree in the future [58]. Recommendations are given to a user based on the evaluation
of items of other users with whom he/she shares common preferences. The first step of the recommendation process is therefore, for each user \( u \), to identify similar users \( v \) through a similarity metric (such as cosine, Jaccard, Pearson, etc.—[59]), \( \text{sim}(u, v) \), and to build a neighborhood for \( u \) with the \( k \) most similar users \( v \). The preference of the user \( u \) for each item \( i \) is then computed as the sum, weighted by \( \text{sim}(u, v) \), of the values of the link weight values (0 or 1) of item \( i \) for the \( k \)-nearest neighbors.

\[
\text{pred}(u, i) = \frac{\sum_{v=1}^{k} \text{sim}(u, v) w_{v,i}}{\sum_{v=1}^{k} \text{sim}(u, v)}
\]  

(9)

where \( w_{v,i} \) is 1 if user \( v \) rated item \( i \) and 0 otherwise. The items with the highest predicted value \( \text{pred}(u, i) \) are recommended to user \( u \).

In our setting, we use a cosine similarity metric to determine the 20 neighbors environment, computed by \( \text{sim}(u, v) = \text{cosine}(u, v) = (\mathbf{v}_u^T \mathbf{v}_v) / (\|\mathbf{v}_u\| \|\mathbf{v}_v\|) \), where each user \( u \) is characterized by a binary vector \( \mathbf{v}_u \) encoding the items that this user saw or consumed (see [59]).

5.1.3. Latent Class Model (LC)

Hofmann and Puzicha [60] introduce a latent variable model called Latent Class Model in the context of collaborative filtering. This clustering model assumes that the preferences of a user are established through a number of latent variables and offers a high degree of flexibility in modeling preference behavior: users may have a variety of specific interests, some of which are shared with some people, and others with others [61]—each hidden preference pattern corresponds to a group of users sharing the same interests and their associated items [62].

In this model, a latent class variable \( z \) is associated with each observation \((u, i)\) where the variable \( u \) represents a user and variable \( i \) represents a liked item. The key assumption made is that \( u \) and \( i \) are independent, conditioned on \( z \). The predicted ratings are then computed based on a probabilistic model using the three variables mentioned above (see [62] for more details on the model and the computational steps).

We set the number of latent classes to 20 for the MovieLens dataset and to 15 for the Book Crossing dataset, as these settings perform well (in terms of accuracy) on these datasets (see [61]).

5.2. Novelty-Oriented Algorithms

5.2.1. Inverted Recommendations (IR)

Instead of using a traditional recommendation process—where we recommend for each user a list of items—that favors the recommendation of popular items, [63] proposed to give most items a fair chance to be recommended to users.

A standard user-based approach (see Equation (9)) first determines, for each user \( u \), a set of \( k \) users (i.e., his/her neighbors) which are the most similar (calculated from a similarity measure) to \( u \). The use of an inverted neighborhood changes the neighborhoods, as all the users \( v \) for which the target user \( u \) is among the \( k \) most similar to \( v \) are now selected as the neighborhood of \( u \) (which thus leads to the fact that the number of neighbors of each user is no longer constant, since it depends on each user). The preference of the user \( u \) for each item \( i \) is still computed by Equation (9), except that the composition of each neighborhood has been adapted.

Using the inverted nearest neighbor’s technique flattens the influence power of users so that all opinions matter in the recommendation process. The proposed method improves the novelty of the recommendation as well as the diversity of the sales while maintaining a good trade-off with the accuracy of the recommendation [63].
5.2.2. Reranking (RR)

Traditionally, recommender systems predict unknown ratings from known ratings (using any recommendation algorithm, e.g., CF, MF, etc.). Items are then recommended to users using a standard ranking approach that ranks the predicted ratings from highest to lowest.

Adomavicius and Kwon proposed an item popularity-based ranking [64] where the ranking criterion is rather based on item popularity—the number of ratings an item received. Recommended items are ranked from lowest to highest popularity. However, using popularity as the only ranking criterion is highly damageable for the relevance of the recommendations and thus the accuracy of the system.

RR is inspired by this parametrized ranking approach proposed in [64], and a ranking threshold $T_R$ is introduced on the predicted score to balance the accuracy and novelty of the recommended list. More precisely, all the items with predicted scores above the threshold $T_R$ are reranked from least to most popular, while items below $T_R$ follow the standard ranking strategy (where the ranking criterion is still the predicted score). With a high $T_R$, accuracy is favored, whereas with a low $T_R$, other dimensions (i.e., novelty) are highlighted.

In our experiment, we applied this procedure to a CF algorithm using half of the maximum of the predicted score as threshold $T_R$, for both datasets.

5.2.3. Exploriometer (XP)

Chatzicharalampous et al. [65] proposed an original neighborhood selection technique aimed at emphasizing the variety of tastes. The exploration habit of a user is the average rarity of his/her rated items (the rarity of an item is estimated by the number of users who have rated it). Mainstream users, who mainly rate popular items, are opposed to domain experts, who tend to explore beyond popular items. The latter are considered useful and favored in the neighborhood constitution. The proposed technique aims at increasing novelty, coverage, and diversity [54].

More precisely, a preliminary step of identifying users who are “explorers” is first computed before building neighborhoods by selecting the explorers who are the most similar (computed from a similarity measure) to the user. In our setting, to be consistent with CF, a 20 neighbors environment is built based on the cosine similarity metric.

6. Results

For each experiment, we applied a 10-fold bootstrapping data-splitting process (either random as a baseline or following a Poisson distribution). We therefore show, in the following tables, scores averaged over the 10 folds of the process. The best results (using a 5% significance level) are shown in bold. Random bootstrapping was chosen as the baseline process (rather than random crossvalidation) for a fair comparison with the new (which is, by definition, forced to follow a bootstrapping process) model (note that a random crossvalidation process has also been applied, and the results, as expected in random processes, are very close to those of random bootstrapping but are not reported in the sequel).

Since our goal is to compare different, state-of-the-art (and thus a priori exclusively accuracy oriented even though they may naturally be more or less novelty oriented) and novelty-oriented (and thus a priori both novelty- and accuracy-oriented) algorithms, we do not show in this Results section raw accuracy or novelty scores but rather relative scores. For each metric (NDCG or EPC$_{X,Y,Z}$), relative scores are obtained by applying a standard normalization:

$$s_{norm}^{a} = \frac{s_{a} - \min(s_{a})}{\max(s_{a}) - \min(s_{a})}$$

where $s_{a}$ is the raw score obtained by algorithm $a$ (e.g., CF) on a particular metric and $\min(s_{a})$ ($\max(s_{a})$) is the minimum (maximum) raw score obtained, for the same dataset, process, and metric, by the worse (best) of the six algorithms. Such a normalization procedure aims
at distributing the scores of the different algorithms between 0 (for the algorithm with the minimum raw score) and 1 (for the algorithm with the maximum raw score).

In addition, we also provide a ranking of all the algorithms. The algorithms are ranked from 1 (the best algorithm on a particular metric) to 6 (the worst algorithm on a particular metric), as we tested six different algorithms. Note that the normalization described above does not trivially affect these rankings.

Analysis

A negative correlation. As expected from the related work analysis (i.e., the introduction of a popularity bias into the evaluation of random data-splitting processes), we first noticed a negative correlation between accuracy and novelty scores in a traditional training and test set split (see Table 3). Typically, algorithms that offer high recommendations accuracy obtain low novelty scores and vice versa.

Table 3. Accuracy (NDCG) and novelty (EPC) scores of the different algorithms when applying a classical random 10-fold bootstrap process, on the BC and ML datasets. The left part of each column contains the normalized (on the six rows of the column) score, while its right part contains the rank (from 1 to 6) of the algorithm (on the six rows of the column).

| Algorithm | Accuracy | EPC |
|-----------|----------|-----|
|           | NDCG 1.1 | 1.1.2 | 1.2.1 | 1.2.2 | 2.1.1 | 2.1.2 | 2.2.1 | 2.2.2 | 3.1 | 3.2 |
| MF        | 1.000 1 | 0.073 5 | 0.063 5 | 0.074 5 | 0.060 5 | 0.038 5 | 0.041 5 | 0.044 5 | 0.047 5 | 0.044 5 | 0.046 5 |
| CF        | 0.716 2 | 0.000 6 | 0.000 6 | 0.000 6 | 0.000 6 | 0.000 6 | 0.000 6 | 0.000 6 | 0.000 6 | 0.000 6 | 0.000 6 |
| LC        | 0.550 3 | 0.211 4 | 0.155 4 | 0.211 4 | 0.149 4 | 0.153 4 | 0.115 4 | 0.178 4 | 0.136 4 | 0.178 4 | 0.135 4 |
| IR        | 0.443 4 | 1.000 1 | 1.000 1 | 1.000 1 | 1.000 1 | 1.000 1 | 1.000 1 | 1.000 1 | 1.000 1 | 1.000 1 | 1.000 1 |
| RR        | 0.226 5 | 0.291 3 | 0.707 3 | 0.292 3 | 0.709 3 | 0.543 2 | 0.228 3 | 0.587 2 | 0.228 3 | 0.586 2 | 0.586 2 |
| XP        | 0.000 6 | 0.709 2 | 0.719 2 | 0.708 2 | 0.716 2 | 0.560 2 | 0.538 3 | 0.602 2 | 0.583 3 | 0.602 2 | 0.582 3 |

More precisely, the best algorithms in terms of accuracy (NDCG) are MF, CF, and LC, for both datasets, while the worst are XP and RR (for both datasets). Note that IR obtains, by far, the best accuracy score of the three novelty-based algorithms, for both datasets. Considering the novelty scores (regardless of the choice of the EPC variant), the best algorithms are IR for BC and RR for ML, while the worst are, by far, MF, LC, and CF (with close but poor results), for both datasets. This negative correlation between accuracy and novelty in traditional evaluation processes shows the difficulty of comparing such algorithms and forms the basis of this work.

The new framework. The purpose of Table 4 is to compare the accuracy scores (i.e., NDCG) of all the algorithms in the traditional framework (see the column “Classical”) versus the new model (i.e., $\lambda = 2, \lambda = 4$ or $\lambda = 6$) integrating novelty and accuracy into a single score. In this new, and more realistic, model, the results show that the most accurate algorithm for $\lambda = 2$ is LC for BC, and MF for ML, and is IR for $\lambda = 4$ and $\lambda = 6$ for BC, and RR for $\lambda = 4$ and $\lambda = 6$ for ML.

Note that, as expected (since the model does not change the algorithm but only the evaluation process) and as highlighted in Table 5 (where each novelty score is computed as the average of all variants of the EPC scores—since all provide very similar scores, as shown in Table 3)—the novelty of the algorithms is weakly impacted by the configurations. A novel
algorithm, in a random configuration, thus remains novel in the $\lambda = 2$, $\lambda = 4$, or $\lambda = 6$ configuration.

Table 4. Evolution of the accuracy scores (i.e., NDCG) of the different algorithms when comparing a classical random 10-fold bootstrap process (see the columns “Classical”) and our new model (see the columns $\lambda = 2$, $\lambda = 4$ and $\lambda = 6$). The left part of each column contains the normalized score (on the six rows of the column), while its right part contains the rank (from 1 to 6) of the algorithm (on the six rows of the column).

| Book-Crossing | MovieLens |
|---------------|-----------|
| Classical     | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 6$ | Classical | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 6$ |
| MF            | 1.000 | 1 | 0.739 | 4 | 0.000 | 6 | 0.000 | 6 | MF | 1.000 | 1 | 1.000 | 1 | 0.317 | 3 | 0.113 | 5 |
| CF            | 0.716 | 2 | 0.889 | 2 | 0.579 | 4 | 0.555 | 4 | CF | 0.913 | 3 | 0.413 | 3 | 0.000 | 6 | 0.000 | 6 |
| LC            | 0.550 | 3 | 1.000 | 1 | 0.319 | 5 | 0.195 | 5 | LC | 0.965 | 2 | 0.909 | 2 | 0.474 | 2 | 0.369 | 3 |
| IR            | 0.443 | 4 | 0.629 | 5 | 1.000 | 1 | 1.000 | 1 | IR | 0.775 | 4 | 0.223 | 4 | 0.281 | 4 | 0.421 | 2 |
| RR            | 0.226 | 5 | 0.838 | 3 | 0.979 | 2 | 0.874 | 2 | RR | 0.000 | 6 | 0.067 | 5 | 1.000 | 1 | 1.000 | 1 |
| XP            | 0.000 | 6 | 0.000 | 6 | 0.643 | 3 | 0.864 | 3 | XP | 0.593 | 5 | 0.000 | 6 | 0.170 | 5 | 0.358 | 4 |

Table 5. Evolution of the novelty scores (i.e., computed as the renormalized average of the scores of all variants of the EPC scores) of the various algorithms when comparing a classical random 10-fold bootstrap process (see the columns “Classical”) and our new framework (see the columns $\lambda = 2$, $\lambda = 4$ and $\lambda = 6$). The left part of each column contains the normalized score (on the six rows of the column), while its right part contains the rank (from 1 to 6) of the algorithm (on the six rows of the column).

| Book-Crossing | MovieLens |
|---------------|-----------|
| Classical     | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 6$ | Classical | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 6$ |
| MF            | 0.044 | 5 | 0.000 | 6 | 0.000 | 6 | 0.000 | 6 | MF | 0.238 | 5 | 0.276 | 5 | 0.313 | 5 | 0.267 | 5 |
| CF            | 0.000 | 6 | 0.229 | 5 | 0.149 | 5 | 0.190 | 5 | CF | 0.000 | 6 | 0.000 | 6 | 0.000 | 6 | 0.000 | 6 |
| LC            | 0.159 | 4 | 0.240 | 4 | 0.198 | 4 | 0.202 | 4 | LC | 0.258 | 4 | 0.313 | 4 | 0.387 | 4 | 0.283 | 4 |
| IR            | 1.000 | 1 | 0.803 | 2 | 0.885 | 2 | 0.823 | 2 | IR | 0.602 | 2 | 0.801 | 2 | 0.932 | 2 | 0.880 | 2 |
| RR            | 0.309 | 3 | 0.304 | 3 | 0.246 | 3 | 0.282 | 3 | RR | 1.000 | 1 | 1.000 | 1 | 1.000 | 1 | 0.650 | 3 |
| XP            | 0.588 | 2 | 1.000 | 1 | 1.000 | 1 | 1.000 | 1 | XP | 0.287 | 3 | 0.548 | 3 | 0.741 | 3 | 1.000 | 1 |

Focusing on the BC dataset (see Table 4), we notice that a shift from a standard split to $\lambda = 2$ (where only highly popular items (i.e., $k = 0$) are penalized with $\lambda = 2$ (while highly popular items, except for the first group, are still strongly favored in the construction of the test set, see Figure 1)) has an impact on the scores of all algorithms. The most originally accurate algorithm (MF)—thus, the one likely to recommend highly popular items—is negatively affected, while all other algorithms, except XP, benefit from the $\lambda = 2$ configuration. The LC classical novelty score (i.e., 0.159, see Table 5), although low, is significantly higher than those of MF and CF (i.e., 0.044 and 0.000), indicating that LC accuracy orientation is achieved by recommending less-but-still-popular items (compared to MF and CF). Moreover, we observe that the accuracy gap between the state-of-the-art and novelty algorithms is significantly reduced (except for XP), meaning that the choice of which algorithm to apply in a real case should not be made solely on a pure accuracy score if novelty, at least some of it, is expected.

Using a $\lambda = 4$ configuration, we observe that originally accurate but non-novelty-oriented algorithms, such as MF, are penalized as their new accuracy positions decrease—MF is the most accurate algorithm in random but ranks sixth in $\lambda = 4$—whereas originally less accurate but novelty-oriented algorithms, such as IR or RR, are favored and see their new accuracy positions increase—IR (RR) originally ranks fourth (fifth) but claims, in $\lambda = 4$, the first (second) position. The $\lambda = 6$ configuration leads to similar results to those obtained with $\lambda = 4$, meaning that the novelty-based algorithms (IR, RR, and XP) are the most accurate algorithms if novelty is expected.

The same conclusions can be drawn for the MovieLens dataset (see Tables 3 and 4) except for the $\lambda = 2$ configuration where the ranking is still the same as in the traditional
framework and that the best algorithm for $\lambda = 4$ and $\lambda = 6$ is now RR, far ahead. We also observe that LC is the most stable algorithm among the three accuracy-oriented algorithms.

Summary. We observe on both datasets a change in the accuracy leading algorithms in our model, which integrates novelty and accuracy into a single score. We also observe that the more highly popular items in the dataset, the faster this change occurs (i.e., when switching from a classical framework through $\lambda = 2$, $\lambda = 4$, and finally $\lambda = 6$ models).

7. Discussions

In recent years, recommender systems have become an important tool in people’s daily lives. To be useful, recommender systems should meet the needs, expectations, and desires of users. The traditional system evaluation (i.e., accuracy oriented) mostly penalizes non-accuracy-oriented systems (e.g., novelty-oriented systems) even though there is a strong correlation between novelty and user preferences [66]. Traditional system evaluations thus potentially prevent the development of non-accuracy-oriented algorithms, and the authors first demonstrate systematically that the gain in novelty is not “too damaging” from an accuracy perspective when developing a novelty-oriented algorithm. Proposing integrated evaluation techniques that do not consider only accuracy seems to be an appropriate solution to promote (systems that focus on) the novelty dimension. However, to the best of our knowledge, there is no work comparing recommender systems with or without the use of traditional training/test set frameworks (which are biased by popularity). Applied to novelty, our goal is to show the evolution of the behavior of state-of-the-art algorithms when novelty is progressively given more importance in the evaluation procedure. If we refer to the related work section (see Section 2.4), a few other papers [19–21,50] propose alternative processes for data splitting, with respect to popularity bias. In the work of Bellogín et al., the approach (divide the data so that all items have the same number of test ratings) was tested, but only accuracy measures were computed and reported (showing that the popularity-neutralising variants yield worse accuracy results—precision and NDCG). In [20], two experiments were conducted (one on the entire datasets and the other only on the long-tail—thus aiming to compare the results obtained using (or not) popular items in the experiment—which is different from the goal of our work, and thus from our framework), and the results show interesting changes in the ranking of the different algorithms that were tested, providing complementary information to our results. Finally, in the work [50], where different values of novelty are used to form test sets, the authors show that the more novel the test sets, the worse the accuracy (precision) results.

Being able to decide on the importance of novelty (and keeping in mind that the optimal degree of novelty depends on various factors such as context, nature of the recommended items, users’ preferences, etc.) is therefore a serious consideration. In our data-splitting process, novelty and accuracy are integrated into a single combined score—depending on a lambda “novelty” parameter—which allows us to compare algorithms adapted to each situation, thus leading to a less artificially biased framework and thus closer to reality.

The results on two well-known datasets show the evolution of the behavior of state-of-the-art algorithms when novelty is progressively given more importance in the evaluation procedure and could lead to potential changes in the decision-making processes of organizations involving recommender systems. Indeed, the proposed methodology could help managers to choose the most suitable algorithm according to the company’s strategy, the objective pursued, etc. (switching, for example and based on our results, from MF or CF algorithms, to IR algorithm if an algorithm that is both accurate and novel is needed), thus facilitating the decision-making process and potentially impacting performance.

8. Conclusions and Future Work

Traditionally, when determining the quality of a recommender system, authors first refer to accuracy metrics, while a novelty score could further be computed—for novelty-oriented algorithms—to show the novelty orientation of the algorithm. However, recom-
mendation algorithms should be compared in a more realistic way, considering simultaneously both accuracy and novelty dimensions.

By using a Poisson distribution to split the data for evaluation—less popular items are more likely (i.e., depending on a parameter) to integrate the test set compared to a uniform distribution—the accuracy metric then includes, to some extent, the novelty dimension and allows for a more realistic comparison between algorithms. The results of this new model show the true face of recommendation algorithms on a dimension combining accuracy and novelty, i.e., in their ability to recommend items which are both accurate and novel/or novel and accurate.

Future work should focus on developing new original set construction techniques to highlight other dimensions such as diversity, serendipity, etc., so that recommendation algorithms could be evaluated in an integrated way. Indeed, developing a “global” data-splitting process that allows any algorithm—regardless of its purpose (i.e., novelty, diversity, serendipity, accuracy, etc.)—to be assessed in a fair and convenient manner would be of great interest.

Author Contributions: Conceptualization, F.F. and E.F.; Investigation, F.F. and E.F.; Methodology, F.F. and E.F.; Validation, F.F. and E.F.; Writing—original draft, F.F. and E.F.; Writing—review and editing, F.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jannach, D.; Resnick, P.; Tuzhilin, A.; Zanker, M. Recommender Systems—Beyond Matrix Completion. Commun. ACM 2016, 59, 94–102. [CrossRef]
2. Jalili, M.; Ahmadian, S.; Izadi, M.; Moradi, P.; Salehi, M. Evaluating Collaborative Filtering Recommender Algorithms: A Survey. IEEE Access 2018, 6, 74003–74024. [CrossRef]
3. Bobadilla, J.; Alonso, S.; Hernando, A. Deep Learning Architecture for Collaborative Filtering Recommender Systems. Appl. Sci. 2020, 10, 2441. [CrossRef]
4. Ricci, F.; Rokach, L.; Shapiro, B.; Kantor, P.B. Recommender Systems Handbook; Springer: Berlin/Heidelberg, Germany, 2011.
5. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutierrez, A. Recommender systems survey. Knowl. Based Syst. 2013, 46, 109–132. [CrossRef]
6. Fouss, F.; Saerens, M. Evaluating Performance of Recommender Systems: An Experimental Comparison. In Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Sydney, Australia, 9–12 December 2008; pp. 735–738. [CrossRef]
7. Shani, G.; Gunawardana, A., Evaluating Recommendation Systems. In Recommender Systems Handbook; Springer: Boston, MA, USA, 2011; pp. 257–297. [CrossRef]
8. Konstan, J.A.; Riedl, J. Recommender systems: From algorithms to user experience. User Model. User Adapt. Interact. 2012, 22, 101–123. [CrossRef]
9. Shafqat, W.; Byun, Y.C. Enabling “Untact” Culture via Online Product Recommendations: An Optimized Graph-CNN based Approach. Appl. Sci. 2020, 10, 5445. [CrossRef]
10. NGuyen, L.; Hong, M.S.; Jung, J.; Sohn, B.S. Cognitive Similarity-Based Collaborative Filtering Recommendation System. Appl. Sci. 2020, 10, 4183. [CrossRef]
11. Valcarce, D.; Bellogín, A.; Parapar, J.; Castells, P. On the Robustness and Discriminative Power of Information Retrieval Metrics for top-N Recommendation. In Proceedings of the 12th ACM Conference on Recommender Systems, Vancouver, BC, Canada, 2 October 2018; pp. 260–268. [CrossRef]
12. del Olmo, F.H.; Gaudioso, E. Evaluation of recommender systems: A new approach. Expert Syst. Appl. 2008, 35, 790–804. [CrossRef]
13. Candillier, L.; Meyer, F.; Boule, M. Comparing state-of-the-art collaborative filtering systems. In Proceedings of the International Workshop on Machine Learning and Data Mining in Pattern Recognition, Leipzig, Germany, 18–20 July 2007; pp. 548–562.
14. Lara-Cabrera, R.; Gonzalez-Prieto, A.; Ortega, F. Deep Matrix Factorization Approach for Collaborative Filtering Recommender Systems. Appl. Sci. 2020, 10, 4926. [CrossRef]
45. Steck, H. Training and Testing of Recommender Systems on Data Missing Not at Random. In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24–28 July 2010; pp. 713–722. [CrossRef]

46. Steck, H. Item Popularity and Recommendation Accuracy. In Proceedings of the Fifth ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; pp. 125–132. [CrossRef]

47. Shi, Y.; Karatzoglou, A.; Baltrunas, L.; Larson, M.; Oliver, N.; Hanjalic, A. CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-more Filtering. In Proceedings of the Sixth ACM Conference on Recommender Systems, Dublin, Ireland, 9–13 September 2012; pp. 139–146. [CrossRef]

48. Said, A.; Bellogín, A. Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. In Proceedings of the 8th ACM Conference on Recommender Systems, Foster City, Silicon Valley, CA, USA, 6–10 October 2014.

49. Gunawardana, A.; Shani, G. A survey of accuracy evaluation metrics of recommendation tasks. J. Mach. Learn. Res. 2009, 10, 2935–2962. [CrossRef]

50. Zhang, M.; Hurley, N. Avoiding Monotony: Improving the Diversity of Recommendation Lists. In Proceedings of the 2008 ACM Conference on Recommender Systems, Lausanne, Switzerland, 23–25 July 2008; pp. 123–130. [CrossRef]

51. Salton, G. Automatic Information Organization and Retrieval; McGraw-Hill: Maidenhead, NY, USA, 1968. [CrossRef]

52. Wang, Y.; Wang, L.; Li, Y.; He, D.; Chen, W.; Liu, T.Y. A Theoretical Analysis of NDCG Ranking Measures. J. Mach. Learn. Res. 2013, 8, 1–30.

53. Breese, J.S.; Heckerman, D.; Kadie, C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI, USA, 24–26 July 1998; pp. 43–52. [CrossRef]

54. Han, J.; Yamana, H. A Survey on Recommendation Methods Beyond Accuracy. IEICE Trans. Inf. Syst. 2017, 100, 2931–2944. [CrossRef]

55. Koren, Y.; Bell, R.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. Computer 2009, 42, 30–37. [CrossRef]

56. Nguyen, H.; Dinh, T. A Modified Regularized Non-Negative Matrix Factorization for MovieLens. In Proceedings of the 2012 IEEE RIVF International Conference on Computing Communication Technologies, Research, Innovation, and Vision for the Future, Ho Chi Minh City, Vietnam, 27 February–1 March 2012; pp. 1–5. [CrossRef]

57. Wu, Z.; Tian, H.; Zhu, X.; Wang, S. Optimization Matrix Factorization Recommendation Algorithm Based on Rating Centrality. In Data Mining and Big Data; Tan, Y., Shi, Y., Tang, Q., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 114–125. [CrossRef]

58. Zhao, X.; Niu, Z.; Chen, W. Opinion-Based Collaborative Filtering to Solve Popularity Bias in Recommender Systems. In Database and Expert Systems Applications; Decker, H., Lhotská, L., Link, S., Basl, J., Tjoa, A.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 426–433.

59. Dunham, M. Data Mining: Introductory and Advanced Topics; Prentice Hall: Hoboken, NJ, USA, 2003.

60. Hofmann, T.; Puzicha, J. Latent Class Models for Collaborative Filtering. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 31 July–6 August 1999; pp. 688–693. [CrossRef]

61. Fouss, F.; Francoisse, K.; Yen, L.; Pirotte, A.; Saerens, M. An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. Neural Netw. 2012, 31, 53–72. [CrossRef] [PubMed]

62. Cheung, K.W.; Tsui, K.C.; Liu, J. Extended latent class models for collaborative recommendation. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. 2004, 34, 143–148. [CrossRef]

63. Vargas, S.; Castells, P. Improving Sales Diversity by Recommending Users to Items. In Proceedings of the 8th ACM Conference on Recommender Systems, Foster City, Silicon Valley, CA, USA, 6–10 October 2014; pp. 145–152. [CrossRef]

64. Adomavicius, G.; Kwon, Y. Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. IEEE Trans. Knowl. Data Eng. 2012, 24, 896–911. [CrossRef]

65. Chatzicharalampous, E.; Christos, Z.; Vakali, A. Exploriometer: Leveraging Personality Traits for Coverage and Diversity Aware Recommendations. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 1463–1468. [CrossRef]

66. Ekstrand, M.D.; Klüver, D.; Harper, F.M.; Konstan, J.A. Letting Users Choose Recommender Algorithms: An Experimental Study. In Proceedings of the 9th ACM Conference on Recommender Systems, Vienna, Austria, 16–20 September 2015; pp. 11–18. [CrossRef]