Top-N Recommendation Algorithms: A Quest for the State-of-the-Art

VITO WALTER ANELLI, Politecnico di Bari, Italy
ALEJANDRO BELLOGÍN, Universidad Autónoma de Madrid, Spain
TOMMASO DI NOIA, Politecnico di Bari, Italy
DIETMAR JANNACH, University of Klagenfurt, Austria
CLAUDIO POMO, Politecnico di Bari, Italy

Research on recommender systems algorithms, like other areas of applied machine learning, is largely dominated by efforts to *improve the state-of-the-art*, typically in terms of accuracy measures. Several recent research works however indicate that the reported improvements over the years sometimes "don't add up", and that methods that were published several years ago often outperform the latest models when evaluated independently. Different factors contribute to this phenomenon, including that some researchers probably often only fine-tune their own models but not the baselines.

In this paper, we report the outcomes of an in-depth, systematic, and reproducible comparison of ten collaborative filtering algorithms—covering both traditional and neural models—on several common performance measures on three datasets which are frequently used for evaluation in the recent literature. Our results show that there is no consistent winner across datasets and metrics for the examined *top-n* recommendation task. Moreover, we find that for none of the accuracy measurements any of the considered neural models led to the best performance. Regarding the performance ranking of algorithms across the measurements, we found that linear models, nearest-neighbor methods, and traditional matrix factorization consistently perform well for the evaluated modest-sized, but commonly-used datasets. Our work shall therefore serve as a guideline for researchers regarding existing baselines to consider in future performance comparisons. Moreover, by providing a set of fine-tuned baseline models for different datasets, we hope that our work helps to establish a common understanding of the state-of-the-art for *top-n* recommendation tasks.

CCS Concepts: • **Information systems → Recommender systems**.

Additional Key Words and Phrases: Recommender Systems, Performance Comparison, Reproducibility

1 INTRODUCTION

Recommender systems are nowadays widely used in online applications, where they help users find relevant information in situations of information overload. Given the high practical relevance of such systems, research in this field is flourishing, particularly in the underlying machine learning (ML) algorithms used to create personalized item suggestions. Correspondingly, the predominant methodology is offline experimentation where the prediction or ranking accuracy of different ML models is compared. The common goal in such research works is to advance the *state-of-the-art*, and evidence is then provided by reporting improvements over existing models that were obtained in those experiments.

Unfortunately, a number of recent research works published in the area of recommender systems and other related areas of applied ML research, e.g., information retrieval, indicate that some of these improvements that have been reported over the years "don't add up" [4]. Ferrari Dacrema et al. [12], for example, benchmark a variety of recent *top-n* recommendation models against earlier and often simpler models. Through their studies, they found that much of the reported progress only

Manuscript submitted to ACM

seems to be "virtual", as the latest models are almost always outperformed by existing methods (see also Rendle et al. [29] for a related analysis). Various reasons may contribute to this surprising phenomenon, including the choice of (too weak) baselines [19, 22] or the lack of a proper tuning of the baselines. Moreover, in such independent evaluations, i.e., that are not done by authors of the compared methods, it often turns out that there is no clear winner across datasets and accuracy measures. Thus, it remains unclear what represents the actual state-of-the-art in this field, given that the ranking of algorithms seems to depend on the particular experimental configuration in terms of baselines, accuracy measures, or datasets.

With this work, our goal is to provide insights regarding what represents the state-of-the-art for *top-n* recommendation tasks, at least for those experimental settings that are common in the recent literature. Like in Ferrari Dacrema et al. [12], we consider a broad range of collaborative filtering algorithms, which includes both older methods based on nearest-neighbors, different matrix factorization approaches, linear models, as well as more recent techniques based on deep learning. Differently from earlier comparisons like Ferrari Dacrema et al. [12], however, we benchmark all algorithms under identical experimental conditions, i.e., with the same datasets and using the same evaluation protocol, after systematically tuning the hyperparameters of all models to reach their best performance.[1]

The outcomes of our experiments show that in none of the considered cases one of the two recent neural methods was the best-performing algorithm. Moreover, the ranking of the algorithms, as expected from the literature, varies across datasets and evaluation measures. With some surprise, we found that linear models, nearest neighbors, and traditional matrix factorization and are dominating the leaderboard across datasets and performance metrics. One insight from our research therefore is that these top-ranking non-neural methods from our analysis should be considered as baselines in future research on recommendation algorithms.

It is worth noticing that the datasets used in our experiments were chosen based on the predominant practice in the current academic literature. In our view, these datasets are however relatively small and different results might be obtained for larger datasets. Such an analysis is however not the focus of our present work, which aims to provide insights on the state-of-the-art in commonly used evaluation setups. Nonetheless, with this work we provide a set of fine-tuned models for these common datasets, thereby reducing the effort for other researchers to tune these baselines in their own experiments. In the future, we plan to publish fine-tuned models also for larger datasets, thereby continuously growing our understanding of the state-of-the-art in this area.

The paper is organized as follows. Next, in Section 2, we describe the details of our methodology and the datasets, algorithms, and metrics that we used in our experiments. Section 3 discusses the outcomes of our experiments, both in terms of accuracy and beyond-accuracy metrics. Section 4 finally discusses and summarizes the insights of our research and provides an outlook on future works.

2 METHODOLOGY

The goal of our study was to evaluate different algorithms under very *common* experimental settings in the current literature in terms of datasets, evaluation metrics, and protocols. The choice of experimental settings reported in this paper were guided by the following considerations. First, we took inspiration from the work by Sun et al. [36], who systematically evaluated various algorithms under a large set of experimental configurations. Second, to select specific experimental configurations for the purpose of our study, we scanned the current literature for the rather common settings. This also led to the inclusion of a number of recent models as well as simpler methods that have proven effective in recent works, where some of them had not been considered in Sun et al..

[1] We share all code and data used to run the experiments publicly to ensure reproducibility of our findings, see our GitHub repository.

Notably, our present work generally differs from Sun et al. in terms of the main goal. In Sun et al., one main purpose was to assess the impact of various aspects of the experimental procedure, e.g., negative sampling, split-ratio, or dataset preprocessing, on accuracy. In contrast, our work mainly focuses on providing a performance comparison of algorithms of different families for very common experimental configurations. Thus, we hope that our work helps establish an agreed-upon and continuously updated benchmark setting that can be used for researchers to test their new models against existing ones in a predefined setting.[2]

2.1 Datasets and Preprocessing

We report the results we obtained for three datasets that are frequently used in the recent literature: *MovieLens-1M*, *Amazon Digital Music*, and *Epinions*.

- *MovieLens-1M* (*ML1M*): The MovieLens datasets have been widely used in the recommender systems literature for many years [13] and different versions are available online[3]. The *ML1M* dataset used in our studies was collected between the years 2000 and 2003 on the MovieLens website and contains ratings for movies on a 1-5 scale. A particularity of the dataset is that it is rather dense, and for each user at least 20 ratings are available.
- *Amazon Digital Music* (*AMZm*): This dataset is part of a larger public collection of datasets[4] that was created initially in the context of image-based recommendation [25]. The Digital Music dataset contains reviews crawled from the Amazon website as well as item ratings on a 1-5 scale.
- *Epinions*: This dataset was crawled in 2003 from the now defunct consumer review site `epinions.com`[5]. A peculiar characteristic of the Epinions website was that users were paid according to how much a review was found useful. For this reason, Epinions has been widely adopted for research on *trust* in recommender systems. The Epinions collection consists of two datasets: one contains item ratings (1-5 stars), while the other one collects (unary) trust statements among users. We point out that, in our study, instead of setting a custom (and in some ways arbitrary) threshold to binarize the rating dataset, we use the second dataset and consider the "trustable" users as the objective of the recommendation task.

It is noteworthy that, for the purpose of our research, all three datasets are publicly available and they were selected also in order to cover a diverse set of application domains of recommender systems. Other datasets, e.g., from the Netflix Prize, were also popular for some time, but they are nowadays only rarely used, e.g., Liang et al. [21], and they are no longer officially accessible. Moreover, differently from the Netflix Prize competition, *rating prediction* is also no longer considered the most important task in recommendation. Instead, the common goal nowadays is to compute *item rankings*. In addition, recommending based on implicit feedback signals is dominating the landscape, given the typical lack of explicit rating information in many applications. Therefore, datasets that originally contain item ratings are commonly converted into unary (like) signals. We follow this practice also in our evaluation and convert the rating datasets of MovieLens and Amazon Digital Music to unary datasets by considering every rating above 3 as a positive signal.[6] For the Epinions dataset, such a conversion is not needed as the data is already given in unary form.

[2] We note that some of the choices regarding the experimental settings could have been made differently as well, e.g., with respect to cutoff thresholds. We however do not expect largely different results when changing some of these minor experiment parameters.
[3] https://grouplens.org/datasets/movielens/
[4] https://jmcauley.ucsd.edu/data/amazon/
[5] http://www.trustlet.org/epinions.html
[6] Alternative approaches exist in the literature for this conversion, e.g., considering every rating as positive in case it is higher than the user's average. Often, we also see that all ratings are converted to positive signals. This is however questionable as *(i)* a low rating, e.g., one star, is not a positive signal and *(ii)* it changes the problem into predicting who will rate what.

Real-world datasets are often very sparse. Therefore, another common pre-processing step in the literature is to create a more dense version of the datasets to ensure that there is a minimum number of interactions per user and item in the dataset, e.g., to allow for effective personalization. We created different *p-cores* for each dataset due to their diverging characteristics. In a *p-core* dataset, we ensure that there are at least p interactions for each item and at least p interactions for each user. For our experiments, the creation of these p-core datasets was done in an iterative procedure, where the described constraints are applied until no more changes to the dataset can be observed. Different values for p were used for the given datasets, depending on their size and density. For Movielens 1M and Amazon Digital Music we used the most common value of p (p=10 for ML1M, p=5 for *AMZm*, see Sun et al. [36]), while for Epinions we choose a p-core value to reach a comparable density of the final matrix with respect to the other two datasets. Specifically, in this latter case, only a 2-core subset was computed due to the dataset's high sparsity. The resulting dataset characteristics are shown in Table 1. We observe that removing negative ratings and creating p-cores led to a considerable reduction of the dataset size for *ML1M*, and that it results in an even more drastic reduction for the *AMZm* dataset.

Table 1. Dataset characteristics before and after pre-processing

Dataset	p-core	#interactions	#users	#items	#interactions	#users	#items
		before pre-processing			*after pre-processing*		
Movielens 1M	10-core	1,000,209	6,040	3,706	571,531	5,949	2,810
Amazon Digital Music	5-core	1,584,082	840,372	456,992	145,523	14,354	10,027
Epinions	2-core	300,548	8,514	8,510†	300,475	8,485	8,463†

†: The Epinions dataset focuses on "trustable" user recommendation. Note that not all users are trustable candidates according to the historical transactions, which is why the number of users as recommendable items is lower than the number of users.

Interestingly, today's commonly used datasets are often not only almost twenty years old, but also rather small, compared, for example, to the Netflix Prize dataset with its 100 million ratings. We assume that the computational complexity of some modern models prevents authors to explore their proposals on larger datasets. Among the larger *public* datasets, the 20M version of the MovieLens datasets is sometimes used in the literature [21]. The 1M version is however used for evaluations more frequently [36], and this is the main reason why we consider it in this study. Moreover, we observed that systematically tuning the hyperparameters for all datasets and models can be computationally challenging for some models already for the datasets of modest size described in Table 1.

2.2 Algorithms

Given the goals described above, we considered algorithms from different families in our analysis. All non-neural methods, except Bayesian Personalized Ranking (BPRMF) [28], were also considered as baselines in the recent analysis of recommendation algorithms presented in Ferrari Dacrema et al. [12]. Specifically, we considered the following techniques in our evaluation:

- Non-personalized baseline: Popularity-based recommendation (*MostPop*).
- Neighborhood-based and simple graph-based models: UserKNN [32], ItemKNN [33], RP$^3\beta$ [27].
- Linear models: SLIM [26], EASE^R [34].
- Matrix factorization models: BPRMF [28], MF2020 [29], iALS [15].
- Neural models: NeuMF [14], MultiVAE [21].

Table 2 provides more details for the compared algorithms and explains why we considered them for our study.

Table 2. Overview of compared algorithms

Family	Algorithm	Description
Non-personalized Baselines	MostPop	Recommends the most popular items to each user, where popularity is defined by the number of observed interactions in the training data.
	Random	Creates random recommendations for users. Mainly useful to provide a reference point for beyond-accuracy measures (see Section 2.3).
Neighbors and Graphs	UserKNN	A user-based nearest neighbor scheme proposed by Resnick et al. [32] in 1994 in an early paper on the GroupLens system. In general, we include early nearest-neighbor techniques here because *(i)* they let us gauge the progress on small datasets over time and *(ii)* they proved surprisingly effective in recent research [12].
	ItemKNN	Item-based nearest-neighbor algorithms were discussed in 2001 [33] and later successfully applied in industry around 2003 [24].
	$RP^3\beta$	This method ($RP^3\beta$) is a simple graph-based method [27] from 2017, which is conceptually similar to the ItemKNN method and can, despite its simplicity, lead to good performance [12].
Linear Models	SLIM	This regression-based method was proposed for top-n recommendation tasks in 2011 [26]. Like in a recent analysis [12], we use the *ElasticNet* version of the method [20], as it often leads to competitive results.
	$EASE^R$	Another linear model, proposed in 2019 [34], which works like a shallow autoencoder. We include this method because it often leads to good results despite its simplicity.
Matrix Factorization	MF2020	Matrix factorization methods were initially explored using Singular Value Decomposition in 1998 [6]. Later, in particular during and after the Netflix Prize, various machine learning approaches were proposed to learn latent factors[7]. A recent analysis shows that these methods from the late 2000s are still competitive. In our study, we use a very recent MF model from Rendle et al. [29] proposed in 2020, dubbed MF2020.
	iALS	This method from 2008 uses an Alternating Least Squares approach and is particularly designed to learn factor models for implicit feedback datasets [15]. The method is widely used as a non-neural baseline in the literature.
	BPRMF	This method from 2009 was also designed for implicit feedback and introduces a novel optimization criterion. We use the MF variant in our experiments, which is also frequently used as a non-neural baseline in the literature [28].
Neural Models	NeuMF	NeuMF was proposed in 2017 [14] and is an early and influential deep learning model used for recommendation. It generalizes matrix factorization and replaces the inner product with a neural architecture. The method is widely used as a neural baseline in the recent literature.
	MultiVAE	This model was designed for implicit feedback data, published in 2018, and is based on variational autoencoders [21]. According to the analysis in Ferrari Dacrema et al. [12], this method outperformed existing non-neural baselines in an independent evaluation.

2.3 Evaluation Settings and Metrics

In this section, we provide details about the applied evaluation protocol, the evaluation metrics, and the hyperparameter tuning process.

[7] https://sifter.org/simon/journal/20061211.html

Evaluation Protocol. We used a common *repeated* 80-20 hold-out splitting procedure in our experiments [29]. Correspondingly, each dataset is randomly split to sample chunks containing around 20% of the data. In each evaluation round, 20% of the data are used for testing and the remaining 80% are for training. Each experiment is repeated five times. Later in Section 3, we report the mean of the observed values of the cross-validation runs.

We note that in the recent literature often only the results of one single training-test split are reported. While this data-splitting is typically done randomly in previous studies, we argue that cross-validation usually leads to more reliable results.

Metrics. We collect a rich variety of accuracy metrics as well as a number of "beyond-accuracy" measures that are commonly used in the literature to assess additional quality aspects of recommendation lists.

- In terms of *accuracy metrics*, we measure Normalized Discounted Cumulative Gain (nDCG), Mean Reciprocal Rank (MRR), Precision, Recall, Mean Average Precision (MAP), and F1 at common list lengths of 10, and 20. For F1, note that we compute it on a per-user basis and not simply as a harmonic mean of the averages of Precision and Recall across users.[8] Thus, we have both metrics that take the position of the correct items into account and metrics that are agnostic of this aspect. Note here that we do not collect "sampled" metrics in our evaluation. In a *sampled metrics* approach, one test item is ranked within an often small list of "negative samples". Such a procedure, while widely used, was recently found to be unreliable [18]. Note that historically the majority of the literature considered error metrics (RMSE, MAE) for evaluation purposes. However, "*such classical error criteria do not really measure top-N performance*" [9]. Consequently, several ranking metrics have been proposed in the last two decades and were adopted to evaluate top-n recommendation tasks. The present work shows the evaluation results for the most commonly used ranking metrics.
- Considering *beyond-accuracy metrics*, we measured a broader range of metrics regarding popularity bias, novelty, fairness, and item coverage and concentration. The details of the considered metrics are provided in Table 3. We note that also the novelty and fairness metrics used here are based on popularity distributions of items. Specifically, for the PRSP and the PREO metrics, we consider the 20% most popular items as the "short head" and the rest as long-tail items.
- *Running times:* Modern machine learning models can be computationally expensive. Therefore, we measured the computation times required for each algorithm for training and testing.

Hyperparameter tuning. We performed extensive hyperparameter tuning for all algorithms in our comparison, which is essential to understand what represents the *state-of-the-art*. Previous research [8] has identified that the lack of proper tuning of baseline algorithms may easily lead to a certain stagnation in the field, where new models are carefully tuned, whereas only limited effort sometimes goes into tuning existing baseline models.

For hyperparameter tuning, we relied on the HyperOpt library[9] and used Tree of Parzen Estimators (TPE) as an algorithm to find the best hyperparameters [5]. We determined suitable hyperparameter ranges for each algorithm from the literature, using, e.g., ranges that were earlier used in Ferrari Dacrema et al. [12] and other works. Depending on the number and ranges of the hyperparameters of each algorithms, we explored between 20 and 50 hyperparameter combinations for each model. Hyperparameter tuning was conducted on a validation set for each dataset, and nDCG@10 was used as an optimization target. As suggested by Anelli et al. [3], the nDCG metric represents a reasonable choice for

[8]With this user-wise calculation of F1, the overall average of F1 values is not bounded to lie between the overall averages of Precision and Recall; see the online material for additional explanations (https://github.com/sisinflab/Top-N-Recommendation-Algorithms-A-Quest-for-the-State-of-the-Art)
[9]http://hyperopt.github.io/hyperopt/

Table 3. Overview of beyond-accuracy metrics

Aspect	Metric	Description
Coverage and Concentration	**IC**	Item Coverage (IC) measures how many items ever appear in the top-n recommendations of users.
	Gini	A measure of statistical dispersion, used to express the inequality of a distribution. A higher Gini index value ($\texttt{Gini} \in [0,\ldots,1]$) indicates a stronger concentration of the recommendations, e.g., on popular items [16]. To ease the interpretation of the results and associate higher values with better results in terms of non-concentrated recommendations, in Tables 9, 10, and 11 we report the value ($1-\texttt{Gini}$).
Novelty	**EFD**	Expected Free Discovery: A novelty measure proposed in [38] based on the *inverse collection frequency*. Like EPC, this measure expresses the ability of an algorithm to recommend relevant long-tail items.
	EPC	Expected Popularity Complement: This metric expresses the *expected "number of seen items not previously seen"* [38].
Fairness	**PREO**	The Popularity-based Ranking-based Equal Opportunity (REO) recommendation metric for assessing bias (fairness) was proposed in [39]. Lower values mean less biased recommendations.
	PRSP	Popularity-based Ranking-based Statistical Parity [39], to assess potential bias and thus fairness of the recommendations. Again, lower values mean less biased recommendations.
Popularity Bias	**APLT**	Average Popularity of Long-Tail Items: Measures the average popularity of long tail items in the top-n recommendations of users [1].
	ARP	Average Rating-based Popularity: This metric computes the popularity of the items in a recommendation list based on the number of interactions of each item in the training data [16].
	ACLT	Average Coverage of Long-Tail Items: Measures how many items from the long tail are covered in the top-n recommendations of users [1].

hyperparameter tuning. All hyperparameter ranges and the optimal values for each dataset and algorithm are reported in the provided online material for reproducibility.

3 RESULTS

3.1 Accuracy Results

The results of the accuracy measurements for commonly used cutoff thresholds of 10 and 20 are shown in Table 4 (MovieLens-1M), Table 5 (Amazon Digital Music), and Table 6 (Epinions). The results for the cutoff threshold of 50 are provided in the online material. We mark the best-performing method for each metric in bold font; the second-best result is underlined. The following main observations can be made.

- *Top-performing methods:* Considering nDCG as our main performance measure—most other metrics are correlated except for Recall in some situations—we find that the top three positions across all metrics and cutoff lengths are taken by five algorithms: EASE^R, MF2020, SLIM, RP$^3\beta$, and, a bit surprisingly, UserKNN. Differences across the datasets exist, but the ranking at least at top places is quite consistent across the datasets. For *ML1M*, EASE^R, MF2020, and SLIM are the best methods, whereas RP$^3\beta$, EASE^R, and SLIM are best for *AMZm*. These methods also work well in Epinions. For the Epinions dataset, however, UserKNN works even slightly better than EASE^R. Generally, the performance of the five top-performing methods is quite consistent, *with EASE^R always taking a top rank*. The MF2020 technique, in contrast, mainly seems to work particularly well for the dense ML1M dataset. We note here that UserKNN

for the given datasets was always favorable over ItemKNN. It is noticeable that this evidence differs from some prior literature. In 2004 [10], it was suggested that item-based algorithms provide comparable or better quality recommendations than traditional user-neighborhood-based recommender systems. In 2011, researchers reported [26] that in their experiments item-based schemes outperform user-based ones. Similar observations were made in 2011 by Ekstrand et al. [11] for rating prediction tasks. In 2016, Christakopoulou and Karypis [7] generally assumed that the item-based methods had been shown to outperform the user-based schemes for the *top-n* recommendation task. In the analysis from 2021 [12], however, a general dominance of ItemKNN over UserKNN was not reported. There were cases where ItemKNN was better, but in the majority of the reported experiments UserKNN was favorable, which suggests that the ranking of the methods may depend on dataset characteristics and specifics of the evaluation protocol.

Table 4. Accuracy Results for MovieLens-1M. The tables are sorted by nDCG in descending order. The notation *@N* indicates that the metrics are computed considering recommendation lists of *N* elements.

Algorithm	Top@10						Algorithm	Top@20					
	nDCG	MAP	MRR	Pre	Rec	F1		nDCG	MAP	MRR	Pre	Rec	F1
EASE^R	**0.336**	0.335	**0.583**	0.274	**0.194**	0.190	EASE^R	**0.335**	0.287	**0.587**	**0.216**	**0.289**	0.206
SLIM	0.335	**0.337**	0.580	**0.275**	0.189	0.188	SLIM	0.332	**0.288**	0.584	**0.216**	0.283	0.204
MF2020	0.329	0.327	0.563	0.272	0.190	**0.192**	MF2020	0.329	0.283	0.568	**0.216**	0.286	**0.207**
UserKNN	0.315	0.314	0.554	0.256	0.183	0.179	RP$^3\beta$	0.315	0.269	0.561	0.203	0.277	0.195
RP$^3\beta$	0.315	0.313	0.556	0.256	0.184	0.179	UserKNN	0.314	0.268	0.559	0.201	0.273	0.192
iALS	0.306	0.304	0.542	0.252	0.179	0.176	iALS	0.309	0.263	0.547	0.202	0.272	0.194
MultiVAE	0.294	0.284	0.514	0.243	0.183	0.175	MultiVAE	0.304	0.250	0.519	0.199	0.281	0.195
ItemKNN	0.292	0.293	0.518	0.242	0.163	0.163	ItemKNN	0.289	0.252	0.523	0.192	0.247	0.180
NeuMF	0.277	0.275	0.494	0.232	0.157	0.158	BPRMF	0.280	0.235	0.508	0.181	0.253	0.176
BPRMF	0.275	0.271	0.502	0.226	0.166	0.161	NeuMF	0.280	0.240	0.500	0.188	0.245	0.195
MostPop	0.159	0.159	0.317	0.137	0.084	0.086	MostPop	0.161	0.141	0.326	0.114	0.137	0.103
Random	0.008	0.007	0.020	0.007	0.004	0.004	Random	0.009	0.007	0.024	0.007	0.007	0.006

Table 5. Accuracy Results for Amazon Digital Music. The tables are sorted by nDCG in descending order.

Algorithm	Top@10						Algorithm	Top@20					
	nDCG	MAP	MRR	Pre	Rec	F1		nDCG	MAP	MRR	Pre	Rec	F1
RP$^3\beta$	**0.085**	**0.040**	**0.115**	**0.023**	0.104	**0.036**	RP$^3\beta$	**0.094**	**0.029**	**0.118**	**0.015**	0.132	**0.026**
EASE^R	0.083	0.038	0.108	**0.023**	**0.106**	0.035	EASE^R	0.092	0.028	0.111	**0.015**	**0.136**	**0.026**
SLIM	0.081	0.037	0.106	0.022	0.104	0.035	SLIM	0.090	0.027	0.109	0.014	0.134	0.025
UserKNN	0.081	0.037	0.105	0.022	0.104	0.035	UserKNN	0.090	0.027	0.108	0.014	0.133	0.025
iALS	0.073	0.032	0.093	0.021	0.099	0.033	iALS	0.084	0.025	0.096	0.014	0.132	0.025
ItemKNN	0.071	0.033	0.095	0.018	0.085	0.029	ItemKNN	0.078	0.023	0.098	0.012	0.108	0.021
MF2020	0.057	0.024	0.067	0.017	0.083	0.028	MF2020	0.067	0.019	0.071	0.012	0.116	0.022
NeuMF	0.056	0.024	0.068	0.015	0.074	0.025	NeuMF	0.063	0.018	0.071	0.010	0.096	0.019
MultiVAE	0.054	0.023	0.062	0.016	0.077	0.025	MultiVAE	0.062	0.017	0.066	0.011	0.105	0.019
BPRMF	0.020	0.008	0.023	0.007	0.031	0.010	BPRMF	0.025	0.007	0.025	0.005	0.047	0.009
MostPop	0.012	0.005	0.016	0.004	0.016	0.006	MostPop	0.014	0.004	0.017	0.003	0.025	0.005
Random	0.000	0.000	0.000	0.000	0.001	0.000	Random	0.001	0.000	0.001	0.000	0.001	0.000

Table 6. Accuracy Results for Epinions. The tables are sorted by nDCG in descending order.

Algorithm	Top@10						Algorithm	Top@20					
	nDCG	MAP	MRR	Pre	Rec	F1		nDCG	MAP	MRR	Pre	Rec	F1
UserKNN	**0.164**	0.131	0.266	0.157	0.102	**0.100**	UserKNN	**0.178**	0.108	0.272	0.076	**0.219**	0.093
EASE^R	**0.164**	**0.132**	**0.268**	0.154	**0.104**	**0.100**	EASE^R	0.177	**0.110**	**0.274**	**0.078**	0.216	**0.094**
RP$^3\beta$	0.163	0.129	0.260	**0.159**	0.102	**0.100**	RP$^3\beta$	0.176	0.107	0.266	0.075	0.218	0.092
SLIM	0.156	0.126	0.254	0.148	0.101	0.096	Slim	0.170	0.106	0.260	0.076	0.210	0.091
MultiVAE	0.149	0.116	0.240	0.148	0.094	0.094	MultiVAE	0.165	0.099	0.247	0.072	0.212	0.089
ItemKNN	0.138	0.111	0.224	0.133	0.090	0.087	ItemKNN	0.151	0.094	0.230	0.068	0.190	0.084
MF2020	0.125	0.104	0.219	0.114	0.084	0.079	MF2020	0.138	0.088	0.225	0.065	0.170	0.079
NeuMF	0.118	0.098	0.206	0.108	0.080	0.075	NeuMF	0.131	0.084	0.212	0.063	0.162	0.075
BPRMF	0.113	0.093	0.200	0.106	0.076	0.071	BPRMF	0.126	0.079	0.207	0.060	0.160	0.072
iALS	0.110	0.091	0.192	0.101	0.074	0.084	iALS	0.121	0.077	0.198	0.057	0.149	0.079
MostPop	0.045	0.037	0.093	0.036	0.029	0.025	MostPop	0.052	0.032	0.100	0.025	0.061	0.029
Random	0.001	0.001	0.003	0.001	0.001	0.001	Random	0.002	0.001	0.003	0.001	0.002	0.001

- *Performance of neural methods:* The two neural methods considered here, NeuMF and MultiVAE, only led to medium performance on these datasets. While MultiVAE performed very well in an earlier comparison with traditional methods [12], we may assume that the modest size of the datasets might limit the power of this method in our experiment to a certain extent, see also the report on the use of deep learning methods at Netflix [35] or the discussions in Jannach et al. [17].
- *Fine-tuning opportunities:* The iALS and BPRMF methods often led to medium to modest performance in this comparison. Recent work indicates that further enhancing and fine-tuning methods like iALS for specific datasets may lead to additional performance improvements [31]. Note, however, that the goal of our work was to assess the performance of different algorithms under equal opportunities, i.e., by using a systematic but generic hyperparameter optimization procedure. Fine-tuning individual algorithms, e.g., by exploring rather uncommon ranges for the size of the latent factors, is of course possible, but not the focus of our work, which is about establishing a set of baselines (state-of-the-art) to consider in future works. Similar considerations apply for the neural methods, which may also be further tuned for individual datasets.

We note that the differences between the top-performing methods are sometimes small, often between one and a few percent. In papers that propose new models, we would therefore commonly expect statistical significance tests. For the evaluations reported in our study, we omit such tests as we have no prior hypotheses regarding which model would "win". Instead, the goal of our work is to provide guidance for researchers about which methods they might want to consider as baselines for comparison. We note that in many published papers no exact details are provided about how the significance tests are applied and prerequisites were validated. Also, in case of per-user comparisons of means, significance at common α-levels may be easy to achieve due to the large sample sizes [23].

Comparing our algorithm ranking with earlier works [12, 36], we find both commonalities and differences. A general commonality of these studies is that more traditional methods, including linear models, matrix factorization, or nearest neighbors frequently take the top positions of the rankings. For example, the innovative combination of Factorization Machines with BPR loss worked particularly well in [36]. Also SLIM and MF were in top positions for some datasets. Differently from our findings, NeuMF more often worked very well for some of the datasets examined in Sun et al. [36]. A competitive performance of NeuMF was also observed in Ferrari Dacrema et al. [12], where it was, however, usually slightly outperformed by various non-neural methods. These differences may be attributed to different causes, including specifics of data-preprocessing and the evaluation procedures[10]. Differently from many earlier works, we apply cross-validation and compute *p-cores* iteratively instead of only filtering "cold" users and items once. Moreover, for some algorithms we explore a larger number of hyperparameter optimization trials than was done in some earlier works.

Finally, to obtain an overall picture of our accuracy results, we applied a Borda count *ranked voting* scheme to aggregate the outcomes of our experiments. To that purpose, we consider each observed ranking for each dataset and metric as a vote. When applying the original Borda count scheme, each candidate (i.e., algorithm) receives more points when it is placed higher in the ranking. In our lists of 12 candidates, the first candidate receives 11 points and the last-ranked candidate 0 points. Applying this scheme across all accuracy measures at list length 10 leads to the ranking shown in Table 7a.[11]

We emphasize that such a rank-based aggregation should be interpreted with great care as it might, for example, favor methods that work particularly well on a set of correlated metrics. In agreement with the analysis presented by Valcarce

[10] In the original paper proposing NeuMF, the authors for example used a *leave-one-out* procedure where only the last item of each user was retained in the test set [14].

[11] The maximum possible value for a method in Table 7a is 198, as we rank 12 algorithms according to 6 metrics for 3 datasets; 198=(12-1) × 1 × 3. For Table 7b and Table 7c, the maximum is correspondingly 33. Although Tables 4 to 6 report rounded values for the sake of clarity, rankings are assessed considering exact metric values.

Table 7. Algorithm ranking based on Borda count at cutoff length 10.

Rank	Algorithm	Count
1	EASE^R	185
2	RP$^3\beta$	169
3	SLIM	160
4	UserKNN	154
5	MF2020	115
6	ItemKNN	99
7	MultiVAE	92
8	iALS	90
9	NeuMF	61
10	BPRMF	45
11	MostPop	18
12	Random	0

(a) Overall

Rank	Algorithm	Count
1	EASE^R	31
2	UserKNN	27
3	RP3beta	27
4	SLIM	27
5	MF2020	19
6	ItemKNN	16
7	MultiVAE	15
8	iALS	13
9	NeuMF	12
10	BPRMF	7
11	MostPop	3
12	Random	0

(b) nDCG

Rank	Algorithm	Count
1	EASE^R	31
2	RP$^3\beta$	29
3	SLIM	26
4	UserKNN	25
5	MF2020	20
6	MultiVAE	17
7	ItemKNN	15
8	iALS	14
9	NeuMF	9
10	BPRMF	9
11	MostPop	3
12	Random	0

(c) Recall

et al. [37], we observed high correlation between ranking metrics and for the same metric using different cutoffs. For example, in that work, when computing the correlation between cutoffs ranging from 5 to 100, the lowest one was 0.9, which still represents a very strong correlation. Because of this, we only considered one threshold for the measurement shown in Table 7a. Another known limitation of the Borda count scheme is that the ranking might change if a candidate is removed from the lists. Despite these limitations, we believe that the Borda count may represent a helpful summarization approach for the experiments in this paper. More fine-grained applications of the Borda count are possible as well to account for such correlations. In Table 7b and Table 7c, we report the Borda count rankings when considering only one specific measure, nDCG@10 and Recall@10, respectively. We select Recall as an example here, because all other metrics are usually more correlated with nDCG than Recall. The analysis in Table 7c actually shows that RP$^3\beta$ and SLIM work particularly well for Recall and are ranked higher than UserKNN for this metric.

3.2 Beyond-Accuracy Results

Table 9 shows the beyond-accuracy metrics results for the MovieLens dataset for the top-10 and top-20 recommendations[12]. The rows in the table are again sorted by accuracy (nDCG). We highlight the best values for each metric, not considering the Random and MostPop baselines, which only serve as reference points. Recommending random items will, for example, lead to high item coverage, but not to many relevant item suggestions.

In our analysis we found that some of our beyond-accuracy can be highly correlated, which is to some extent expected as many of them are based on item popularity characteristics, as discussed above. Table 8 shows the outcomes of an analysis of metric correlations. In this table, we report in how many cases (datasets) a metric is correlated with another one with a Pearson product-moment correlation coefficient (PPMCC) higher than 0.9 or lower than -0.9. We can observe that both the ACLT and the PRSP metrics are consistently correlated with the APLT metric. For the sake of conciseness, we therefore only report the APLT metric here and omit ACLT and PRSP from the tables. All detailed results also for these metrics can be found in the online material.

[12]Detailed results for other datasets and cutoff thresholds can be found in the online material.

Table 8. Summary of Metric Correlations. A ✓ in a cell indicates a correlation of more than 0.9 (or beyond -0.9 vice versa) for one of the datasets. Two or three ✓ symbols mean that such a high correlation was also found for the second or the third dataset.

PPMCC	Gini	EFD	EPC	PREO	PRSP	ACLT	APLT	ARP
IC	✓	–	–	✓	–	–	–	✓
Gini		–	–	✓	✓✓	✓	✓	–
EFD			✓	–	✓	✓	✓	–
EPC				✓	–	✓	✓	–
PREO					✓	✓	✓	–
PRSP						✓✓✓	✓✓✓	–
ACLT							✓✓✓	–
APLT								–

Table 9. Beyond Accuracy Results for MovieLens-1M. The tables are sorted by nDCG in descending order. The notation *@N* indicates that the metrics are computed considering recommendation lists of *N* elements. To ease the interpretation of the results and to associate higher values with more diversified recommendation lists, we report the value of $1-Gini$.

Algorithm	Top@10							Algorithm	Top@20						
	IC	Gini	EFD	EPC	PREO	APLT	ARP		IC	Gini	EFD	EPC	PREO	APLT	ARP
EASE^R	838.0	0.068	**2.690**	**0.583**	0.978	0.003	1,062.727	EASE^R	1093.0	0.091	**2.264**	**0.207**	0.963	0.006	949.860
SLIM	654.2	0.052	2.672	0.244	0.995	0.001	1,121.384	SLIM	854.0	0.069	2.239	0.205	0.986	0.003	1,017.374
MF2020	920.2	0.077	2.672	0.244	0.968	0.005	1,042.373	MF2020	1128.8	0.095	2.257	**0.207**	0.946	0.009	969.260
UserKNN	1075.2	0.067	2.489	0.227	0.971	0.010	1,085.550	RP$^3\beta$	1207.2	0.073	2.085	0.190	0.937	0.018	1,024.408
RP$^3\beta$	854.4	0.048	2.461	0.223	0.959	0.011	1,181.638	UserKNN	1465.8	0.092	2.090	0.191	0.947	0.017	970.255
iALS	712.2	0.080	2.516	0.232	0.997	0.000	935.914	iALS	901.000	0.105	2.138	0.197	0.983	0.002	838.660
MultiVAE	**1625.2**	**0.136**	2.422	0.221	**0.828**	**0.042**	**871.869**	MultiVAE	**1924.8**	**0.156**	2.082	0.190	**0.792**	**0.052**	**821.849**
ItemKNN	1054.8	0.066	2.346	0.214	0.952	0.011	1,090.926	ItemKNN	1346.0	0.082	1.977	0.181	0.939	0.015	1,006.978
NeuMF	1367.2	0.111	2.292	0.209	0.910	0.028	938.861	BPRMF	1386.2	0.111	1.893	0.173	0.890	0.017	968.784
BPRMF	1137.8	0.091	2.226	0.203	0.928	0.010	1,047.232	NeuMF	1679.4	0.135	1.965	0.179	0.867	0.038	868.452
MostPop	56.2	0.005	1.187	0.103	1.000	0.000	1,746.694	MostPop	92.0	0.010	1.039	0.092	1.000	0.000	1,570.672
Random	2810.0	0.876	0.074	0.006	0.039	0.696	151.045	Random	2810.0	0.911	0.075	0.007	0.037	0.693	151.352

Table 10. Beyond Accuracy Results for Amazon Digital Music. The tables are sorted by nDCG in descending order.

Algorithm	Top@10							Algorithm	Top@20						
	IC	Gini	EFD	EPC	PREO	APLT	ARP		IC	Gini	EFD	EPC	PREO	APLT	ARP
RP$^3\beta$	**9959.0**	**0.542**	**0.409**	**0.033**	0.308	0.299	23.759	RP$^3\beta$	**10016.0**	**0.609**	**0.293**	**0.024**	0.318	0.299	22.353
EASE^R	7789.0	0.178	0.368	0.031	0.537	0.054	56.155	EASE^R	9441.0	0.233	0.267	0.022	0.542	0.071	47.987
SLIM	8215.4	0.197	0.361	0.030	0.552	0.067	49.287	SLIM	9659.4	0.253	0.263	0.022	0.548	0.086	43.337
UserKNN	7703.8	0.181	0.363	0.030	0.552	0.056	51.910	UserKNN	9294.2	0.237	0.264	0.022	0.543	0.073	45.834
iALS	4516.2	0.136	0.325	0.027	0.766	0.009	41.936	iALS	5941.6	0.177	0.243	0.020	0.700	0.015	37.520
ItemKNN	9686.2	0.478	0.345	0.027	**0.097**	**0.550**	**9.884**	ItemKNN	9976.2	0.528	0.247	0.019	**0.121**	**0.546**	**9.870**
MF2020	4722.8	0.099	0.242	0.021	0.687	0.009	59.949	MF2020	6389.4	0.135	0.187	0.016	0.661	0.014	51.479
NeuMF	7365.2	0.228	0.245	0.020	0.455	0.058	30.236	NeuMF	8533.2	0.266	0.180	0.015	0.468	0.075	27.416
MultiVAE	6043.0	0.189	0.235	0.020	0.578	0.045	40.475	MultiVAE	9161.4	0.329	0.178	0.015	0.519	0.094	33.590
BPRMF	3050.0	0.024	0.078	0.007	0.784	0.001	130.810	BPRMF	4283.0	0.034	0.064	0.006	0.787	0.002	108.296
MostPop	15.6	0.001	0.039	0.004	1.000	0.000	182.800	MostPop	29.2	0.002	0.033	0.004	1.000	0.000	148.838
Random	10025.8	0.852	0.002	0.000	0.229	0.460	9.840	Random	10025.8	0.895	0.002	0.000	0.158	0.460	9.838

Generally, we observe that the ranking of the algorithms is not entirely consistent across the datasets. Here, we summarize a number of patterns that we observed, having in mind that beyond-accuracy measures are only of secondary interest in this study.

- For **ARP**, which reports the average item popularity in the top-n lists, we find that BPRMF often has the strongest tendency to recommend popular items on all datasets. MF2020 and EASE^R are also often at the higher end regarding the popularity bias. The ranking of the algorithms however varies across datasets. On the *ML1M* dataset, the

Table 11. Beyond Accuracy Results for Epinions. The tables are sorted by nDCG in descending order.

Algorithm	Top@10						
	IC	Gini	EFD	EPC	PREO	APLT	ARP
UserKNN	3402.2	0.073	**1.198**	0.112	0.398	0.080	315.724
EASER	2765.0	0.055	1.197	**0.114**	0.501	0.046	341.486
RP$^3\beta$	**6009.0**	**0.197**	1.237	0.112	0.198	**0.275**	226.165
SLIM	3361.0	0.081	1.197	0.110	0.452	0.061	246.216
MultiVAE	3386.8	0.089	1.105	0.102	0.364	0.105	293.641
ItemKNN	5832.8	0.160	1.084	0.097	**0.171**	0.267	**214.681**
MF2020	1235.2	0.028	0.921	0.090	0.786	0.008	368.003
NeuMF	3595.0	0.084	0.905	0.085	0.495	0.096	322.865
BPRMF	1322.8	0.018	0.824	0.081	0.651	0.012	475.629
iALS	1613.4	0.064	0.891	0.081	0.707	0.010	214.989
MostPop	41.6	0.001	0.273	0.030	1.000	0.000	719.301
Random	8443.0	0.823	0.011	0.001	0.142	0.738	27.565

Algorithm	Top@20						
	IC	Gini	EFD	EPC	PREO	APLT	ARP
UserKNN	4594.6	0.095	0.962	0.090	0.442	0.083	275.265
EASER	3705.0	0.071	**0.965**	**0.091**	0.522	0.047	297.062
RP$^3\beta$	7010.8	**0.232**	0.983	0.089	0.262	**0.268**	**198.240**
SLIM	4401.2	0.098	**0.965**	0.089	0.493	0.063	228.155
MultiVAE	4249.6	0.112	0.900	0.083	0.426	0.112	255.457
ItemKNN	**7100.2**	0.188	0.875	0.079	**0.272**	0.262	199.488
MF2020	1631.2	0.040	0.766	0.074	0.776	0.009	320.867
NeuMF	4647.0	0.108	0.756	0.071	0.542	0.105	276.817
BPRMF	1793.6	0.026	0.693	0.067	0.640	0.012	409.832
iALS	2052.0	0.078	0.730	0.066	0.654	0.016	202.718
MostPop	72.0	0.002	0.244	0.027	1.000	0.000	629.745
Random	8443.6	0.875	0.011	0.001	0.079	0.738	27.654

differences between algorithms are also generally smaller than for other datasets. On the other end of the spectrum, we observe that the neural methods NeuMF and MultiVAE sometimes succeed to include less popular items in the recommendation lists. RP$^3\beta$ and ItemKNN are similarly successful on the Epinions and *AMZm* in this respect. The **APLT** metric, which considers the popularity and coverage of long-tail items are negatively correlated with the **ARP** metric, i.e., the more popular items are recommended, the fewer from the long tail.

- The novelty metrics **EPC** and **EFD**, like all remaining beyond-accuracy metrics considered here, are generally negatively correlated with the **ARP** metric as well. An interesting pattern here is that models that perform well on the nDCG are also mostly highly ranked in terms of the novelty metrics.
- Looking at the fairness metric **PREO**, which is also based on item popularity and where lower values are better, the picture is not so clear. The neural MultiVAE method, for example, seems to rather consistently produce relatively fair recommendations according to this metric. ItemKNN leads to very good results on the Epinions and Amazon dataset, and to average performance on the *ML1M* dataset. For this latter dataset, the spread of values is however not too high.
- Finally, considering the **Gini** index, MultiVAE generally leads to lower concentration levels on *ML1M*, and ItemKNN and RP$^3\beta$ have lower concentration effects for the Epinions and *AMZm* datasets. Looking at **Item Coverage**, both nearest-neighbor methods and the neural approaches are typically better than the matrix factorization techniques iALS and BPRMF. The patterns are however not consistent across datasets. EASE^R, for example, leads to relatively high item coverage on *AMZm*, but not on the other datasets.

Overall, not many consistent patterns regarding beyond-accuracy measures across all three datasets can be observed. One example of such a pattern is a certain popularity bias of the BPRMF method, which was previously observed [16]. Some patterns, like good item coverage for ItemKNN, are only found for the *AMZm* and Epinions datasets, which suggests that the widely used *ML1M* dataset may be to some extent unique and it stands to question how representative this dense dataset is for other typical application scenarios, e.g., for e-commerce settings.

3.3 Time Measurements

We carried out all experiments on a computing cluster of our organization. The used cluster is based on IBM Power9 processors and has 980 nodes. Each node is equipped with 32 cores and 4 NVIDIA Volta GPUs. One cluster node with 200GB of RAM with 4 logical CPUs was reserved for each experiment. In addition, one NVIDIA Volta GPU with 16GB of RAM has been allocated for the experiments with the neural models NeuMF and MultiVAE. Table 12 shows the time measurements obtained for the three datasets, using the optimal parameters (e.g., number of latent factors) that were

determined through hyperparameter tuning. The numbers reported in the table refer to the time needed (in seconds) to train the model once, and to create and evaluate the recommendation lists for all users in the test set.

Table 12. Training and evaluation time

Algorithm	time (sec.)
MF2020	1.53×10^4
NeuMF	7.97×10^3
BPRMF	3.97×10^3
iALS	331.93
UserKNN	87.29
$EASE^R$	85.93
SLIM	73.19
MultiVAE	67.03
$RP^3\beta$	47.06
ItemKNN	42.74
Random	27.49
MostPop	24.63

(a) MovieLens-1M

Algorithm	time (sec.)
NeuMF	3.57×10^4
iALS	2.90×10^4
MF2020	2.65×10^4
MultiVAE	1.37×10^4
$EASE^R$	1.85×10^3
BPRMF	1.51×10^3
SLIM	403.29
$RP^3\beta$	270.78
ItemKNN	257.96
UserKNN	247.68
Random	50.86
MostPop	45.58

(b) Amazon Digital Music

Algorithm	time (sec.)
iALS	3.27×10^4
MF2020	1.97×10^4
NeuMF	3.46×10^3
BPRMF	2.26×10^3
$EASE^R$	1.12×10^3
SLIM	344.69
MultiVAE	215.43
$RP^3\beta$	148.24
UserKNN	144.05
ItemKNN	139.42
Random	47.99
MostPop	44.24

(c) Epinions

The results show that there is a substantial spread between the algorithms. While there are some models that complete training and testing within one minute, training the MF2020 method on the ML1M dataset, where it performed well, took several days. We note here that more efficient implementations of matrix factorization techniques have been proposed [30]. Also the NeuMF model needed substantial time to complete the computations. In contrast, the MultiVAE model, which was also originally evaluated on larger datasets in Liang et al. [21] was among the fastest models. The neighborhood-based models and $RP^3\beta$ were also implemented for high efficiency. For the other datasets, Epinions and Amazon, the results are similar with NeuMF and the matrix factorization models often taking substantial computation time. For this latter class of models, the efficiency also largely depends on the optimal number of latent factors.

Generally, combining the timing results with accuracy results from above, we see no clear indication for the given datasets that computationally more complex models are favorable in terms of prediction accuracy.

4 SUMMARY, DISCUSSION & OUTLOOK

In recent years, several researchers have identified major challenges with respect to reproducibility and progress in recommender systems research. Various factors contribute to these phenomena, in particular *(a)* that a larger fraction of published research is not reproducible because authors do not share the required artifacts and *(b)* that the experiments in published research mainly aim to highlight the superiority of a new model. In the context of this latter aspect, this practically often means that only the new method is carefully fine-tuned but not the compared baseline methods. Furthermore, the choice of the baselines is sometimes limited to very recent models, thus probably missing strong baselines that were published earlier.

With this work, our goal is to address these open issues in different ways. First, we conducted a large number of reproducible experiments on different datasets and involving a variety of algorithms from different families in order to provide an independent evaluation of existing techniques along different quality and performance measures. The outcomes of these experiments shall help guide researchers in the choice of baseline algorithms to consider in their own research. In particular we found that one should consider algorithms of different types in any evaluation, as there appears to be no single method that is better than all others in all experimental configurations. Second, we ran these experiments with the help of a recent general evaluation framework for recommender systems [2], thus allowing other researchers to benchmark their new models within a defined environment and against already well-tuned baselines.

In terms of the outcomes of the experiments, our reproducibility study confirmed earlier findings that the latest models are not often the best performing ones, in particular for the modest-sized datasets that we considered in our evaluation. In our ongoing and future work, we plan to fine-tune our models also on larger datasets and to share these tuned models publicly. Thereby, we hope to reduce the often huge computational effort that other researchers would otherwise need to fine-tune all baseline models whenever they propose a new model. Over time, this collection of fine-tuned models for various datasets may represent a step towards a shared understanding of what represents the "state-of-the-art" in algorithms research. For these larger datasets, we also expect a more consistent and strong performance of deep learning models.

Besides accuracy metrics, our experiments included a number of beyond-accuracy metrics relating to popularity bias, novelty, fairness, and item coverage. Our results confirm earlier findings that there can be substantial differences between algorithms, e.g., in terms of their tendency to recommend popular items. Such algorithm tendencies can be of high relevance in practical application settings, e.g., when the goal is to support item discovery through the recommendations. An important observation in our research is that common metrics for novelty and fairness are tightly coupled and correlated with general popularity biases[13]. Future research might therefore strive to find alternative metrics that more often go beyond popularity as indicators for novelty, diversity, fairness, or serendipity.

In addition to this, a careful analysis on the effect of the optimization goals for hyperparameter tuning is missing in the literature. The results presented herein considered methods optimized for one specific accuracy-oriented metric, i.e., nDCG. But what would happen if other metrics are used for this optimization? It is true that there are strong correlations between some metrics, as discussed before, but it is also well-known that accuracy and beyond-accuracy measurements are typically inversely related, hence, the question of what "state-of-the-art" means in terms of these other metrics remains open and should be addressed in the future.

Finally, another aspect regarding the splitting strategy has to be taken into consideration. Here, we adopted a random hold-out splitting strategy with repeated experiments that became popular in recent literature. Together with k-folds cross-validation, they are representative of the evaluation protocols adopted in recent works. Nevertheless, random-based splitting strategies undoubtedly favor some methods since information regarding the future general users' behavior is exploited in the training phase. More realistic time-aware splitting strategies should be investigated to study how much they impact the overall ranking of recommendation systems.

ACKNOWLEDGMENTS

We acknowledge the CINECA award under the ISCRA initiative, for the availability of high performance computing resources and support. The authors acknowledge partial support of the projects: PID2019-108965GB-I00, Fincons CdP3, PASSPARTOUT, Servizi Locali 2.0, ERP4.0, Secure Safe Apulia.

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing Popularity Bias in Recommender Systems with Personalized Re-Ranking. In *FLAIRS '19*. 413–418.

[2] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. 2021. Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation. In *The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21*. 2405–2414.

[3] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Claudio Pomo, and Azzurra Ragone. 2019. On the discriminative power of hyper-parameters in cross-validation and how to choose them. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk (Eds.). ACM, 447–451.

[13] In theory, the Gini index is not necessarily tied to popularity biases, but with the typical long-tail distributions it usually captures a concentration of items in the "short head".

[4] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. 2009. Improvements That Don't Add Up: Ad-hoc Retrieval Results Since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. 601–610.

[5] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11)*. 2546–2554.

[6] Daniel Billsus and Michael J. Pazzani. 1998. Learning Collaborative Information Filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*. 46–54.

[7] Evangelia Christakopoulou and George Karypis. 2016. Local Item-Item Models For Top-N Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 67–74.

[8] Paolo Cremonesi and Dietmar Jannach. 2021. Progress in recommender systems research: Crisis? What crisis? *AI Magazine* 42, 3 (2021), 43–54.

[9] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*. ACM, 39–46.

[10] Mukund Deshpande and George Karypis. 2004. Item-based top-*N* recommendation algorithms. *ACM Trans. Inf. Syst.* 22, 1 (2004), 143–177.

[11] Michael D. Ekstrand, Michael Ludwig, Joseph A. Konstan, and John Riedl. 2011. Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit. In *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011*. 133–140.

[12] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Transactions on Information Systems (TOIS)* 39 (2021). Issue 2.

[13] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2015).

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 173–182.

[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*. 263–272.

[16] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. 2015. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *UMUAI* 25, 5 (2015), 427–491.

[17] Dietmar Jannach, Gabriel De Souza P. Moreira, and Even Oldridge. 2020. Why Are Deep Learning Models Not Consistently Winning Recommender Systems Competitions Yet?. In *ACM RecSys Challenge Workshop*. Online.

[18] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. 1748––1757.

[19] Sara Latifi, Noemi Mauro, and Dietmar Jannach. 2021. Session-aware Recommendation: A Surprising Quest for the State-of-the-art. *Information Sciences* 573 (2021), 291–315.

[20] Mark Levy and Kris Jack. 2013. Efficient top-n recommendation by linear regression. In *RecSys Large Scale Recommender Systems Workshop*.

[21] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018*. 689–698.

[22] Jimmy Lin. 2019. The neural hype and comparisons against weak baselines. *ACM SIGIR Forum* 52, 2 (2019), 40–51.

[23] Mingfeng Lin, Henry C. Lucas, and Galit Shmueli. 2013. Research Commentary—Too Big to Fail: Large Samples and the p-Value Problem. *Information Systems Research* 24, 4 (2013), 906–917.

[24] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Comput.* 7, 1 (2003), 76–80.

[25] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. 43–52.

[26] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *11th IEEE International Conference on Data Mining, ICDM 2011*. 497–506.

[27] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. 2017. Updatable, Accurate, Diverse, and Scalable Recommendations for Interactive Applications. *ACM Trans. Interact. Intell. Syst.* 7, 1 (2017), 1:1–1:34.

[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal*. 452–461.

[29] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys '20)*.

[30] Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. 2021. iALS++: Speeding up Matrix Factorization with Subspace Optimization. *CoRR* abs/2110.14044 (2021). arXiv:2110.14044 https://arxiv.org/abs/2110.14044

[31] Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. 2021. Revisiting the Performance of iALS on Item Recommendation Benchmarks. *CoRR* abs/2110.14037 (2021). arXiv:2110.14037 https://arxiv.org/abs/2110.14037

[32] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94)*. 175–186.

[33] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. 285–295.

[34] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *The World Wide Web Conference, WWW 2019*. 3251–3257.

[35] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. 2021. Deep Learning for Recommender Systems: A Netflix Case Study. *AI Magazine* 42, 3 (2021), 7–18.

[36] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. In *Fourteenth ACM Conference on Recommender Systems*. 23–32.

[37] Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. 2018. On the robustness and discriminative power of information retrieval metrics for top-N recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018*. 260–268.

[38] Saúl Vargas and Pablo Castells. 2011. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. 109–116.

[39] Ziwei Zhu, Jianling Wang, and James Caverlee. 2020. Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 449–458.

APPENDIX

A INTER-METRIC CORRELATIONS - BEYOND ACCURACY METRICS

In this section we report the correlations between each pair of beyond-accuracy metrics. For each dataset, the tables indicate the Pearson product-moment correlation coefficient, unveiling strong direct and inverse correlations. Please note we do not report same analysis for accuracy metrics here, since the topic of correlation among those metrics has been extensively studied in prior literature. Please refer to Valcarce et al. [37], and Anelli et al. [3] for further details.

Table 13. Detailed Metric Correlations. The tables show how much each beyond-accuracy metric (computed on recommendation lists of ten items for each user) correlates with each other. Specifically, the table shows the Pearson product-moment correlation coefficient for each dataset.

Movielens	**EFD**	**Gini**	**IC**	**PopREO**	**PopRSP**	**ACLT**	**APLT**	**ARP**
EPC	1.00	-0.73	-0.36	0.75	0.79	-0.79	-0.79	0.20
EFD		-0.74	-0.37	0.77	0.81	-0.80	-0.80	0.23
Gini			0.86	-0.99	-0.99	0.99	0.99	-0.81
IC				-0.86	-0.79	0.80	0.80	-0.95
PopREO					0.99	-0.99	-0.99	0.78
PopRSP						-1.00	-1.00	0.74
ACLT							1.00	-0.74
APLT								-0.74
Amazon	**EFD**	**Gini**	**IC**	**PopREO**	**PopRSP**	**ACLT**	**APLT**	**ARP**
EPC	1.00	-0.10	0.45	-0.26	0.09	-0.04	-0.04	-0.46
EFD		-0.05	0.49	-0.32	0.03	0.02	0.02	-0.50
Gini			0.78	-0.85	-0.96	0.88	0.88	-0.69
IC				-0.93	-0.74	0.70	0.70	-0.88
PopREO					0.87	-0.87	-0.87	0.84
PopRSP						-0.97	-0.97	0.60
ACLT							1.00	-0.58
APLT								-0.58
Epinions	**EFD**	**Gini**	**IC**	**PopREO**	**PopRSP**	**ACLT**	**APLT**	**ARP**
EPC	0.83	-0.05	-0.11	-0.94	-0.84	0.91	0.91	-0.81
EFD		-0.54	-0.59	-0.62	-1.00	0.97	0.97	-0.67
Gini			1.00	-0.25	0.55	-0.43	-0.43	-0.18
IC				-0.19	0.60	-0.49	-0.49	-0.12
PopREO					0.63	-0.74	-0.74	0.81
PopRSP						-0.99	-0.99	0.64
ACLT							1.00	-0.69
APLT								-0.69

B F1 SCORES - ADDITIONAL NUMERICAL EXAMPLES

The F1 score represents the harmonic mean of Precision and Recall. In the recommendation domain, when evaluating lists of k items (top-k evaluation), it is usually defined as follows:

$$F1\ Score = \frac{1}{|U|} \sum_{u \in U} 2 * \frac{P_u@k * R_u@k}{P_u@k + R_u@k} \tag{1}$$

where U is the set of the users in the population, and where $Pu@k$ and $Ru@k$ are the Precision and Recall values for a user u's *top-k* recommendations, respectively. In an alternative formulation, the F1 Score could be computed *after* obtaining the average Precision and Recall values across all users:

$$P@k = \frac{1}{|U|} \sum_{u \in U} P_u@k \tag{2}$$

$$R@k = \frac{1}{|U|} \sum_{u \in U} R_u@k \tag{3}$$

$$F1\ Score = 2 * \frac{P@k * R@k}{P@k + R@k} \tag{4}$$

These alternative formulations may lead to different results, as we highlight in the following examples. Let us consider a population of five users for whom we have computed the Precision and Recall values for a recommendation system A (see Table 14a).

Table 14. Accuracy results for the toy recommendation systems. P@k, R@k, and F@k stands for individual Precision, Recall, and F1 Score with a list of k recommendations, respectively. *Average* reports the overall Precision and Recall values. Per-user F1 and average-based F1 indicates the F1 scores computed using Equation 1 and Equation 4, respectively.

Population	$P_u@k$	$R_u@k$	$F_u@k$
$user_0$	0.2	0.3	0.240
$user_1$	0.5	0.6	0.545
$user_2$	0.3	0.4	0.343
$user_3$	0.6	0.3	0.400
$user_4$	0.2	0.3	0.240
	$P@k$	$R@k$	$F@k$
Average	0.36	0.38	
Per-user F1			0.354
Average-based F1			0.370

(a) Toy recommendation system A.

Population	$P_u@k$	$R_u@k$	$F_u@k$
$user_0$	0.2	0.4	0.267
$user_1$	0.5	0.2	0.286
$user_2$	0.4	0.4	0.400
$user_3$	0.2	0.6	0.300
$user_4$	0.5	0.4	0.444
	$P@k$	$R@k$	$F@k$
Average	0.36	0.40	
Per-user F1			0.339
Average-based F1			0.379

(b) Toy recommendation system B.

It is worth noticing that the F1 formulation from Equation 1, denoted as *Per-User F1*, returns an F1 score that is lower than the overall averaged values of Precision and Recall. This can happen due to the product of individual Precision and Recall values. If one of the two is small, it affects the result and impacts the F1 score. Conversely, this behavior is not likely to occur when the F1 is computed on already averaged Precision and Recall values (Average-based F1).

Furthermore, suppose that we evaluate the performance of two recommender systems, A and B (Table 14b). The two systems lead to the same average Precision value, and B leads to a higher Recall value than A. It may now be surprising to see that A has a higher *per-user* F1 score than B. As a consequence of the previously discussed phenomenon, it is indeed

possible. That is, although the Precision value of system B is equal to system A, some individual Precision values lead to poor individual F1 results that affect the overall value of the metric. Some examples of such cases can be found in the accuracy results of the paper.

C HYPERPARAMETERS RANGE

Table 15. Hyperparameter values for our baselines.

Algorithm	Hyperparameter	Range	Type	Distribution
UserKNN, ItemKNN	topK	5 - 1000	Integer	uniform
	similarity	cosine, jaccard, dice, pearson, euclidean	Categorical	
$RP^3\beta$	topK	5 - 1000	Integer	uniform
	alpha	0 - 2	Real	uniform
	beta	0 - 2	Real	uniform
	normalization	True, False	Categorical	
SLIM	topK	5 -1000	Integer	uniform
	l1 ratio	0.00001 - 1	Real	log-uniform
	alpha	0.01 - 1	Real	uniform
$EASE^R$	l2 norm	1 - 10000000	Real	log-uniform
MF2020	num factors	8, 16, 32, 64, 128, 256	Integer	
	epochs	30 - 100	Integer	uniform
	learning rate	0.00001 - 1	Real	log-uniform
	reg	0.00001 - 0.1	Real	log-uniform
	negative sample	4,6,8	Integer	
iALS	num factors	1 - 200	Integer	uniform
	scaling	linear, log	Categorical	
	alpha	0.001 - 50	Real	uniform
	epsilon	0.001 - 10	Real	uniform
	reg	0.001 - 0.01	Real	uniform
BPRMF	num factors	8, 16, 32, 64, 128, 256	Integer	
	learning rate	0.00001 - 1	Real	log-uniform
	batch size	128, 256, 512	Integer	
	reg user	0.00001 - 0.1	Real	log-uniform
	reg positive item	0.00001 - 0.1	Real	log-uniform
	reg negative item	0.00001 - 0.1	Real	log-uniform
NeuMF	num factors	8, 16, 32, 64, 128, 256	Integer	
	epochs	30 - 100	Integer	uniform
	learning rate	0.00001 - 1	Real	log-uniform
	batch size	128, 256, 512	Integer	
	negative sample	4,6,8	Integer	
MultiVAE	epochs	100 - 300	Integer	uniform
	learning rate	0.00001 - 1	Real	log-uniform
	batch_size	64, 128, 256	Integer	
	intermediate dim	400 - 800	Integer	uniform
	latent dim	100-400	Integer	uniform
	reg	0.00001 - 1	Real	log-uniform

Table 16. Hyperparameter values for our baselines on all datasets.

Algorithm	Hyperparameter	MovieLens	Amazon	Epinions
UserKNN	topK	117	226	139
	similarity	correlation	cosine	cosine
ItemNN	topK	95	798	137
	similarity	cosine	cosine	cosine
$RP^3\beta$	topK	158	803	144
	alpha	1.4350197	0.4973207	0.8719344
	beta	0.3265517	0.2836938	0.2483698
	normalization	true	false	true
SLIM	topK	518	663	663
	l1 ratio	0.0000420	0.0000108	0.0000108
	alpha	0.2978543	0.0486771	0.0486771
$EASE^R$	l2 norm	238.5621338	238.5621338	238.5621338
MF2020	num factors	128	64	16
	epochs	72	92	97
	learning rate	0.1295965	0.1295965	0.0154435
	reg	0.0087583	0.0125009	0.0223642
	negative sample	4	8	4
iALS	num factors	51	200	178
	epochs	27	70	145
	scaling	log	log	log
	alpha	6.3818930	9.1219718	2.8537184
	epsilon	5.6496278	0.4921936	2.3098481
	reg	0.0494734	0.4921936	0.0411491
BPRMF	num factors	256	64	256
	epochs	73	86	63
	learning rate	0.0378936	0.1265624	0.1004075
	batch size	256	256	256
	reg user	0.0157839	0.0058673	0.0002613
	reg positive item	0.0005651	0.0052985	0.0034511
	reg negative item	0.0012779	0.0009577	0.0328127
NeuMF	num factors	16	128	32
	epochs	93	100	39
	learning rate	0.0000366	0.0001365	0.0000465
	batch size	256	64	256
	negative sample	6	6	8
MultiVAE	epochs	100	205	200
	learning rate	0.0001545	0.0000723	0.0001003
	batch_size	128	128	128
	intermediate dim	674	721	674
	latent dim	175	279	175
	reg	0.0000105	0.1153400	0.0020018