Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches

Maurizio Ferrari Dacrema
Politecnico di Milano, Italy
maurizio.ferrari@polimi.it

Paolo Cremonesi
Politecnico di Milano, Italy
paolo.cremonesi@polimi.it

Dietmar Jannach
University of Klagenfurt, Austria
dietmar.jannach@aau.at

ABSTRACT

Deep learning techniques have become the method of choice for researchers working on algorithmic aspects of recommender systems. With the strongly increased interest in machine learning in general, it has, as a result, become difficult to keep track of what represents the state-of-the-art at the moment, e.g., for top-n recommendation tasks. At the same time, several recent publications point out problems in today's research practice in applied machine learning, e.g., in terms of the reproducibility of the results or the choice of the baselines when proposing new models.

In this work, we report the results of a systematic analysis of algorithmic proposals for top-n recommendation tasks. Specifically, we considered 18 algorithms that were presented at top-level research conferences in the last years. Only 7 of them could be reproduced with reasonable effort. For these methods, it however turned out that 6 of them can often be outperformed with comparably simple heuristic methods, e.g., based on nearest-neighbor or graph-based techniques. The remaining one clearly outperformed the baselines but did not consistently outperform a well-tuned non-neural linear ranking method. Overall, our work sheds light on a number of potential problems in today's machine learning scholarship and calls for improved scientific practices in this area.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; **Recommender systems**; • **General and reference** → **Evaluation**.

KEYWORDS

Recommender Systems; Deep Learning; Evaluation; Reproducibility

1 INTRODUCTION

Within only a few years, deep learning techniques have started to dominate the landscape of algorithmic research in recommender systems. Novel methods were proposed for a variety of settings and algorithmic tasks, including top-n recommendation based on long-term preference profiles or for session-based recommendation scenarios [36]. Given the increased interest in machine learning in general, the corresponding number of recent research publications, and the success of deep learning techniques in other fields like vision or language processing, one could expect that substantial progress resulted from these works also in the field of recommender systems. However, indications exist in other application areas of machine learning that the achieved progress—measured in terms of accuracy improvements over existing models—is not always as strong as expected.

Lin [25], for example, discusses two recent neural approaches in the field of information retrieval that were published at top-level conferences. His analysis reveals that the new methods do *not* significantly outperform existing baseline methods when these are carefully tuned. In the context of recommender systems, an in-depth analysis presented in [29] shows that even a very recent neural method for session-based recommendation can, in most cases, be outperformed by very simple methods based, e.g., on nearest-neighbor techniques. Generally, questions regarding the true progress that is achieved in such applied machine learning settings are not new, nor tied to research based on deep learning. Already in 2009, Armstrong et al. [2] concluded from an analysis in the context of ad-hoc retrieval tasks that, despite many papers being published, the reported improvements "don't add up".

Different factors contribute to such phenomena, including (i) weak baselines; (ii) establishment of weak methods as new baselines; and (iii) difficulties in comparing or reproducing results across papers. One first problem lies in the choice of the baselines that are used in the comparisons. Sometimes, baselines are chosen that are too weak in general for the given task and dataset, and sometimes the baselines are not properly fine-tuned. Other times, baselines are chosen from the same family as the newly proposed algorithm, e.g., when a new deep learning algorithm is compared only against other deep learning baselines. This behaviour enforces the propagation of weak baselines. When previous deep learning algorithms were evaluated against too weak baselines, the new deep learning algorithm will not necessarily improve over strong non-neural baselines. Furthermore, with the constant flow of papers being published in recent years, keeping track of what represents a state-of-the-art baseline becomes increasingly challenging.

Besides issues related to the baselines, an additional challenge is that researchers use various types of datasets, evaluation protocols, performance measures, and data preprocessing steps, which makes it difficult to conclude which method is the best across different application scenarios. This is in particular problematic when source code and data are not shared. While we observe an increasing trend

that researchers publish the source code of their algorithms, this is not the common rule today even for top-level publication outlets. And even in cases when the code is published, it is sometimes incomplete and, for instance, does not include the code for data pre-processing, parameter tuning, or the exact evaluation procedures, as pointed out also in [15].

Finally, another general problem might lie in today's research practice in applied machine learning in general. Several "troubling trends" are discussed in [27], including the thinness of reviewer pools or misaligned incentives for authors that might stimulate certain types of research. Earlier work [46] also discusses the community's focus on abstract accuracy measures or the narrow focus of machine learning research in terms of what is "publishable" at top publication outlets.

With this research work, our goal is to shed light on the question if the problems reported above also exist in the domain of deep learning-based recommendation algorithms. Specifically, we address two main research questions:

(1) *Reproducibility*: To what extent is recent research in the area reproducible (with reasonable effort)?
(2) *Progress*: To what extent are recent algorithms actually leading to better performance results when compared to relatively simple, but well-tuned, baseline methods?

To answer these questions, we conducted a systematic study in which we analyzed research papers that proposed new algorithmic approaches for top-n recommendation tasks using deep learning methods. To that purpose, we scanned the recent conference proceedings of KDD, SIGIR, TheWebConf (WWW), and RecSys for corresponding research works. We identified 18 relevant papers.

In a first step, we tried to reproduce the results reported in the paper for those cases where the source code was made available by the authors and where we had access to the data used in the experiments. In the end, we could reproduce the published results with an acceptable degree of certainty for only 7 papers. A first contribution of our work is therefore an assessment of the reproducibility level of current research in the area.

In the second part of our study, we re-executed the experiments reported in the original papers, but also included additional baseline methods in the comparison. Specifically, we used heuristic methods based on user-based and item-based nearest neighbors as well as two variants of a simple graph-based approach. Our study, to some surprise, revealed that in the large majority of the investigated cases (6 out of 7) the proposed deep learning techniques did not consistently outperform the simple, but fine-tuned, baseline methods. In one case, even a non-personalized method that recommends the most popular items to everyone was the best one in terms of certain accuracy measures. Our second contribution therefore lies in the identification of a potentially more far-reaching problem related to current research practices in machine learning.

The paper is organized as follows. Next, in Section 2, we describe our research method and how we reproduced existing works. The results of re-executing the experiments while including additional baselines are provided in Section 3. We finally discuss the implications of our research in Section 4.

2 RESEARCH METHOD

2.1 Collecting Reproducible Papers

To make sure that our work is not only based on individual examples of recently published research, we systematically scanned the proceedings of scientific conferences for relevant long papers in a manual process. Specifically, we included long papers in our analysis that appeared between 2015 and 2018 in the following four conference series: KDD, SIGIR, TheWebConf (WWW), and RecSys.[1] We considered a paper to be relevant if it (a) proposed a deep learning based technique and (b) focused on the top-n recommendation problem. Papers on other recommendation tasks, e.g., group recommendation or session-based recommendation, were not considered in our analysis. Given our interest in top-n recommendation, we considered only papers that used for evaluation classification or ranking metrics, such as Precision, Recall, MAP. After this screening process, we ended up with a collection of 18 relevant papers.

In a next step, we tried to reproduce[2] the results reported in these papers. Our approach to reproducibility is to rely as much as possible on the artifacts provided by the authors themselves, i.e., their source code and the data used in the experiments. In theory, it should be possible to reproduce published results using only the technical descriptions in the papers. In reality, there are, however, many tiny details regarding the implementation of the algorithms and the evaluation procedure, e.g., regarding data splitting, that can have an impact on the experiment outcomes [39].

We therefore tried to obtain the code and the data for all relevant papers from the authors. In case these artifacts were not already publicly provided, we contacted all authors of the papers and waited 30 days for a response. In the end, we considered a paper to be *reproducible*, if the following conditions were met:

- A working version of the *source code* is available or the code only has to be modified in minimal ways to work correctly.[3]
- At least one *dataset* used in the original paper is available. A further requirement here is that either the originally-used train-test splits are publicly available or that they can be reconstructed based on the information in the paper.

Otherwise, we consider a paper to be *non-reproducible* given our specific reproduction approach. Note that we also considered works to be non-reproducible when the source code was published but contained only a skeleton version of the model with many parts and details missing. Concerning the datasets, research based solely on non-public data owned by companies or data that was gathered in some form from the web but not shared publicly, was also not considered reproducible.

The fraction of papers that were reproducible according to our relatively strict criteria per conference series are shown in Table 1.

Overall, we could reproduce only about one third of the works, which confirms previous discussions about limited reproducibility,

---

[1]All of the conferences are either considered A* in the Australian Core Ranking or specifically dedicated to research in recommender systems.
[2]Precisely speaking, we used a mix of replication and reproduction [12, 35], i.e., we used both artifacts provided by the authors and our own artifacts. For the sake of readability, we will only use the term "reproducibility" in this paper.
[3]We did not apply modifications to the core algorithms.

Table 1: Reproducible works on deep learning algorithms for top-n recommendation per conference series from 2015 to 2018.

Conference	Rep. ratio	Reproducible
KDD	3/4 (75%)	[17], [23], [48]
RecSys	1/7 (14%)	[53]
SIGIR	1/3 (30%)	[10]
WWW	2/4 (50%)	[14], [24]
Total	7/18 (39%)	

Non-reproducible: KDD: [43], RecSys: [41], [6], [38], [44], [21], [45], SIGIR: [32], [7], WWW: [42], [11]

see, e.g., [3]. The sample size is too small to make reliable conclusions regarding the difference between conference series. The detailed statistics per year—not shown here for space reasons—however indicate that the reproducibility rate increased over the years.

2.2 Evaluation Methodology

Measurement Method. The validation of the progress that is achieved through new methods against a set of baselines can be done in at least two ways. One is to evaluate all considered methods within the same defined environment, using the same datasets and the exact same evaluation procedure for all algorithms as done in [29]. While such an approach helps us obtain a picture of how different methods compare across datasets, the implemented evaluation procedure might be slightly different from the one used in the original papers. As such, this approach would not allow us to *exactly* reproduce what has been originally reported, which is the goal in this present work.

In this work, we therefore reproduce the work by refactoring the original implementations in a way that allows us to apply the same evaluation procedure that was used in the original papers. Specifically, refactoring is done in a way that the original code for training, hyper-parameter optimization and prediction are separated from the evaluation code. This evaluation code is then also used for the baselines.

For all reproduced algorithms considered in the individual experiments, we used the optimal hyper-parameters that were reported by the authors in the original papers for each dataset. This is appropriate because we used the same datasets, algorithm implementation, and evaluation procedure as in the original papers.[4] We share all the code and data used in our experiments as well as details of the final algorithm (hyper-)parameters of our baselines along with the full experiment results online. [5]

Baselines. We considered the following baseline methods in our experiments, all of which are conceptually simple.

TopPopular: A non-personalized method that recommends the most popular items to everyone. Popularity is measured by the number of explicit or implicit ratings.

---

[4]We will re-run parameter optimization for the reproduced algorithms as part of our future work in order to validate the parameter optimization procedures used by the authors. This step was, however, outside the scope of our current work.
[5]https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation

ItemKNN: A traditional Collaborative-Filtering (CF) approach based on k-nearest-neighborhood (KNN) and item-item similarities [49]. We used the cosine similarity s_{ij} between items i and j computed as

$$s_{ij} = \frac{\mathbf{r}_i \cdot \mathbf{r}_j}{\|\mathbf{r}_i\|\|\mathbf{r}_j\| + h} \tag{1}$$

where vectors $\mathbf{r}_i, \mathbf{r}_j \in \mathbb{R}^{|U|}$ represent the implicit ratings of a user for items i and j, respectively, and $|U|$ is the number of users. Ratings can be optionally weighted either with TF-IDF or BM25, as described in [50]. Furthermore the similarity may or not be normalized via the product of vector norms. Parameter h (the *shrink term*) is used to lower the similarity between items having only few interactions [5]. The other parameter of the method is the neighborhood size k.

UserKNN: A neighborhood-based method using collaborative user-user similarities. Hyper-parameters are the same as used for ItemKNN [40].

ItemKNN-CBF: A neighborhood content-based-filtering (CBF) approach with item similarities computed by using item content features (attributes)

$$s_{ij} = \frac{\mathbf{f}_i \cdot \mathbf{f}_j}{\|\mathbf{f}_i\|\|\mathbf{f}_j\| + h} \tag{2}$$

where vectors $\mathbf{f}_i, \mathbf{f}_j \in \mathbb{R}^{|F|}$ describe the features of items i and j, respectively, and $|F|$ is the number of features. Features can be optionally weighted either with TF-IDF or BM25. Other parameters are the same used for ItemKNN [28].

ItemKNN-CFCBF: A hybrid CF+CFB algorithm based on item-item similarities. The similarity is computed by first concatenating, for each item i, the vector of ratings and the vector of features – $[\mathbf{r}_i , w\mathbf{f}_i]$ – and by later computing the cosine similarity between the concatenated vectors. Hyper-parameters are the same used for ItemKNN, plus a parameter w that weights the content features with respect to the ratings.

P$^3\alpha$: A simple graph-based algorithm which implements a random walk between users and items [8]. Items for user u are ranked based on the probability of a random walk with three steps starting from user u. The probability p_{ui} to jump from user u to item i is computed from the implicit user-rating-matrix as $p_{ui} = (r_{ui}/N_u)^\alpha$, where r_{ui} is the rating of user u on item i, N_u is the number of ratings of user u and α is a damping factor. The probability p_{iu} to jump backward is computed as $p_{iu} = (r_{ui}/N_i)^\alpha$, where N_i is the number of ratings for item i. The method is equivalent to a KNN item-based CF algorithm, with the similarity matrix defined as

$$s_{ij} = \sum_v p_{jv} p_{vi} \tag{3}$$

The parameters of the method are the numbers of neighbors k and the value of α. We include this algorithm because it provides good recommendation quality at a low computational cost.

RP$^3\beta$: A version of P$^3\alpha$ proposed in [34]. Here, the outcomes of P$^3\alpha$ are modified by dividing the similarities by each item's popularity raised to the power of a coefficient β. If β is 0, the algorithm is equivalent to P$^3\alpha$. Its parameters are the numbers of neighbors k and the values for α and β.

For all baseline algorithms and datasets, we determined the optimal parameters via Bayesian search [1] using the implementation

of Scikit-Optimize[6]. We explored 35 cases for each algorithm, where the first 5 were used for the initial random points. We considered neighborhood sizes k from 5 to 800; the shrink term h was between 0 and 1000; and α and β took real values between 0 and 2.

3 VALIDATION AGAINST BASELINES

This section summarizes the results of comparing the reproducible works with the described baseline methods. We share the detailed statistics, results, and final parameters online.

3.1 Collaborative Memory Networks (CMN)

The CMN method was presented at SIGIR '18 and combines memory networks and neural attention mechanisms with latent factor and neighborhood models [10]. To evaluate their approach, the authors compare it with different matrix factorization and neural recommendation approaches as well as with an ItemKNN algorithm (with no *shrinkage*). Three datasets are used for evaluation: Epinions, CiteULike-a, and Pinterest. Optimal hyper-parameters for the proposed method are reported, but no information is provided on how the baselines are tuned. Hit rate and NDCG are the performance measures used in a leave-one-out procedure. The reported results show that CMNs outperform all other baselines on all measures.

We were able to reproduce their experiments for all their datasets. For our additional experiments with the simple baselines, we optimized the parameters of our baselines for the hit rate (HR@5) metric. The results for the three datasets are shown in Table 2.

Our analysis shows that, after optimization of the baselines, CMN[7] is in no single case the best-performing method on any of the datasets. For the CiteULike-a and Pinterest datasets, at least two of the personalized baseline techniques outperformed the CMN method on any measure. Often, even all personalized baselines were better than CMN. For the Epinions dataset, to some surprise, the unpersonalized TopPopular method, which was not included in the original paper, was better than all other algorithms by a large margin. On this dataset, CMN was indeed much better than our baselines. The success of CMN on this comparably small and very sparse dataset with about 660k observations could therefore be tied to the particularities of the dataset or to a popularity bias of CMN. An analysis reveals that the Epinions dataset has indeed a much more uneven popularity distribution than the other datasets (Gini index of 0.69 vs. 0.37 for CiteULike-a). For this dataset, CMN also recommends in its top-n lists items that are, on average, 8% to 25% more popular than the items recommended by our baselines.

3.2 Metapath based Context for RECommendation (MCRec)

MCRec [17], presented at KDD '18, is a meta-path based model that leverages auxiliary information like movie genres for top-n recommendation. From a technical perspective, the authors propose a priority-based sampling technique to select higher-quality path instances and propose a novel co-attention mechanism to improve the representations of meta-path based context, users, and items.

The authors benchmark four variants of their method against a variety of models of different complexity on three small datasets

---

[6]https://scikit-optimize.github.io/
[7]We report the results for CMN-3 as the version with the best results.

Table 2: Experimental results for the CMN method using the metrics and cutoffs reported in the original paper. Numbers are printed in bold when they correspond to the best result or when a baseline outperformed CMN.

	CiteULike-a			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1803	0.1220	0.2783	0.1535
UserKNN	**0.8213**	**0.7033**	**0.8935**	**0.7268**
ItemKNN	**0.8116**	**0.6939**	0.8878	**0.7187**
P$^3\alpha$	**0.8202**	**0.7061**	**0.8901**	**0.7289**
RP$^3\beta$	**0.8226**	**0.7114**	**0.8941**	**0.7347**
CMN	0.8069	0.6666	0.8910	0.6942
	Pinterest			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1668	0.1066	0.2745	0.1411
UserKNN	**0.6886**	**0.4936**	0.8527	**0.5470**
ItemKNN	**0.6966**	**0.4994**	**0.8647**	**0.5542**
P$^3\alpha$	0.6871	**0.4935**	0.8449	**0.5450**
RP$^3\beta$	**0.7018**	**0.5041**	**0.8644**	**0.5571**
CMN	0.6872	0.4883	0.8549	0.5430
	Epinions			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	**0.5429**	**0.4153**	**0.6644**	**0.4547**
UserKNN	0.3506	0.2983	0.3922	0.3117
ItemKNN	0.3821	0.3165	0.4372	0.3343
P$^3\alpha$	0.3510	0.2989	0.3891	0.3112
RP$^3\beta$	0.3511	0.2980	0.3892	0.3103
CMN	0.4195	0.3346	0.4953	0.3592

(MovieLens100k, LastFm, and Yelp). The evaluation is done by creating 80/20 random training-test splits and by executing 10 of such evaluation runs. The evaluation procedure could be reproduced; public training-test splits were provided only for the MovieLens dataset. For the *MF* and *NeuMF* [14] baselines used in their paper, the architecture and hyper-parameters were taken from the original papers; no information about hyper-parameter tuning is provided for the other baselines. Precision, Recall, and the NDCG are used as performance measures, with a recommendation list of length 10. The NDCG measure is however implemented in an uncommon and questionable way, which is not mentioned in the paper. Here, we therefore use a standard version of the NDCG.

In the publicly shared software, the meta-paths are hard-coded for MovieLens, and no code for preprocessing and constructing the meta-paths is provided. Here, we therefore only provide the results for the MovieLens dataset in detail. We optimized our baselines for Precision, as was apparently done in [17]. For MCRec, the results for the complete model are reported.

Table 3 shows that the traditional ItemKNN method, when configured correctly, outperforms MCRec on all performance measures.

Besides the use of an uncommon NDCG measure, we found other potential methodological issues in this paper. Hyper-parameters for the *MF* and *NeuMF* baselines were, as mentioned, not optimized

Table 3: Comparing MCRec against our baselines (Movie-Lens100k)

	PREC@10	REC@10	NDCG@10
TopPopular	0.1907	0.1180	0.1361
UserKNN	0.2913	0.1802	0.2055
ItemKNN	**0.3327**	**0.2199**	**0.2603**
$P^3\alpha$	0.2137	0.1585	0.1838
$RP^3\beta$	0.2357	0.1684	0.1923
MCRec	0.3077	0.2061	0.2363

for the given datasets but taken from the original paper [17]. In addition, looking at the provided source code, it can be seen that the authors report the best results of their method for each metric across different epochs chosen on the test set, which is inappropriate.[8]

3.3 Collaborative Variational Autoencoder (CVAE)

The CVAE method [23], presented at KDD '18, is a hybrid technique that considers both content as well as rating information. The model learns deep latent representations from content data in an unsupervised manner and also learns implicit relationships between items and users from both content and ratings.

The method is evaluated on two comparably small CiteULike datasets (135k and 205k interactions). For both datasets, a sparse and a dense version is tested. The baselines in [23] include three recent deep learning models and as well as Collaborative Topic Regression (CTR). The parameters for each method are tuned based on a validation set. Recall at different list lengths (50 to 300) is used as an evaluation measure. Random train-test data splitting is applied and the measurements are repeated five times.

Table 4: Experimental results for CVAE (CiteULike-a).

	REC@50	REC@100	REC@300
TopPopular	0.0044	0.0081	0.0258
UserKNN	0.0683	0.1016	0.1685
ItemKNN	**0.0788**	0.1153	0.1823
$P^3\alpha$	**0.0788**	0.1151	0.1784
$RP^3\beta$	**0.0811**	0.1184	0.1799
ItemKNN-CFCBF	**0.1837**	**0.2777**	**0.4486**
CVAE	0.0772	0.1548	0.3602

We could reproduce their results using their code and evaluation procedure. The datasets are also shared by the authors. Fine-tuning our baselines led to the results shown in Table 4 for the dense CiteULike-a dataset from [47]. For the shortest list length of 50, even the majority of the pure CF baselines outperformed the CVAE method on this dataset. At longer list lengths, the hybrid *ItemKNN-CFCBF* method led to the best results. Similar results were obtained for the sparse *CiteULike-t* dataset. Generally, at list length 50, *ItemKNN-CFCBF* was consistently outperforming CVAE in all tested configurations. Only at longer list lengths (100 and beyond), CVAE was able to outperform our methods on two datasets.

---

[8]In our evaluations, we did not use this form of measurement.

Overall, CVAE was only favorable over the baselines in certain configurations and at comparably long and rather uncommon recommendation cutoff thresholds. The use of such long list sizes was however not justified in the paper.

3.4 Collaborative Deep Learning (CDL)

The discussed CVAE method considers the earlier and often-cited CDL method [48] from KDD '15 as one of their baselines, and the authors also use the same evaluation procedure and CiteULike datasets. CDL is a probabilistic feed-forward model for joint learning of stacked denoising autoencoders (SDAE) and collaborative filtering. It applies deep learning techniques to jointly learn a deep representation of content information and collaborative information. The evaluation of CDL in [48] showed that it is favorable in particular compared to the widely referenced CTR method [47], especially in sparse data situations.

Table 5: Experimental results for CDL on the dense *CiteULike-a* dataset.

	REC@50	REC@100	REC@300
TopPopular	0.0038	0.0073	0.0258
UserKNN	**0.0685**	0.1028	0.1710
ItemKNN	**0.0846**	**0.1213**	0.1861
$P^3\alpha$	**0.0718**	**0.1079**	0.1777
$RP^3\beta$	**0.0800**	**0.1167**	0.1815
ItemKNN-CBF	**0.2135**	**0.3038**	**0.4707**
ItemKNN-CFCBF	**0.1945**	**0.2896**	**0.4620**
CDL	0.0543	0.1035	0.2627

We reproduced the research in [48], leading to the results shown in Table 5 for the dense *CiteULike-a* dataset. Not surprisingly, the baselines that were better than CVAE in the previous section are also better than CDL, and again for short list lengths, already the pure CF methods were better than the hybrid CDL approach. Again, however, CDL leads to higher Recall for list lengths beyond 100 in two out of four dataset configurations. Comparing the detailed results for CVAE and CDL, we see that the newer CVAE method is indeed always better than CDL, which indicates that progress was made. Both methods, however, are not better than one of the simple baselines in the majority of the cases.

3.5 Neural Collaborative Filtering (NCF)

Neural network-based Collaborative Filtering [14], presented at WWW '17, generalizes Matrix Factorization by replacing the inner product with a neural architecture that can learn an arbitrary function from the data. The proposed hybrid method (NeuMF) was evaluated on two datasets (MovieLens1M and Pinterest), containing 1 million and 1.5 million interactions, respectively. A leave-one out procedure is used in the evaluation and the original data splits are publicly shared by the authors. Their results show that NeuMF is favorable, e.g., over existing matrix factorization models, when using the hit rate and the NDCG as an evaluation measure using different list lengths up to 10.

Parameter optimization is done on a validation set created from the training set. Similar to the implementation of *MCRec* above,

the provided source code shows that the authors chose the number of epochs based on the results obtained for the test set. Since the number of epochs, however, is a parameter to tune and should not be determined based on the test set, we use a more appropriate implementation that finds this parameter with the validation set. For the ItemKNN method, the authors only varied the neighborhood sizes but did not test other variations.

Table 6: Experimental results for NCF.

| | Pinterest | | | |
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1663	0.1065	0.2744	0.1412
UserKNN	0.7001	**0.5033**	0.8610	**0.5557**
ItemKNN	**0.7100**	**0.5092**	**0.8744**	**0.5629**
$P^3\alpha$	0.7008	**0.5018**	0.8667	**0.5559**
$RP^3\beta$	**0.7105**	**0.5116**	0.8740	**0.5650**
NeuMF	0.7024	0.4983	0.8719	0.5536
	Movielens 1M			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.3043	0.2062	0.4531	0.2542
UserKNN	0.4916	0.3328	0.6705	0.3908
ItemKNN	0.4829	0.3328	0.6596	0.3900
$P^3\alpha$	0.4811	0.3331	0.6464	0.3867
$RP^3\beta$	0.4922	0.3409	0.6715	0.3991
NeuMF	0.5486	0.3840	0.7120	0.4369
SLIM	**0.5589**	**0.3961**	**0.7161**	**0.4470**

Given the publicly shared information, we could reproduce the results from [14]. The outcomes of the experiment are shown in Table 6. On the Pinterest dataset, two of the personalized baselines were better than NeuMF on all metrics. For the MovieLens dataset, NeuMF outperformed our simple baselines quite clearly.

Since the MovieLens dataset has been extensively used over the last decades for evaluating new models, we made additional experiments with *SLIM*, a simple linear method described in [33]. To implement *SLIM*, we took the standard Elastic Net implementation provided in the scikit-learn package for Python (ElasticNet). To tune the hyper-parameters on the validation set, we considered neighborhood sizes as in the other baselines; the ratio of l1 and l2 regularization between 10^{-5} and 1.0; and the regularization magnitude coefficient between 10^{-3} and 1.0. Table 6 shows that *SLIM* is indeed better than our baselines, as expected, but also outperforms NeuMF on this dataset.

3.6 Spectral Collaborative Filtering (SpectralCF)

SpectralCF [53], presented at RecSys '18, was designed to specifically address the cold-start problem and is based on concepts of Spectral Graph Theory. Its recommendations are based on the bipartite user-item relationship graph and a novel convolution operation, which is used to make collaborative recommendations directly in the *spectral domain*. The method was evaluated on three public datasets (MovieLens1M, HetRec, and Amazon Instant Video) and

benchmarked against a variety of methods, including recent neural approaches and established factorization and ranking techniques. The evaluation was based on randomly created 80/20 training-test splits and using Recall and the Mean Average Precision (MAP) at different cutoffs.[9]

For the MovieLens dataset, the training and test datasets used by the authors were shared along with the code. For the other datasets, the data splits were not published therefore we created the splits by ourself following the descriptions in the paper.

Somehow surprisingly, the authors report only one set of hyper-parameter values in the paper, which they apparently used for all datasets. We therefore ran the code both with the provided hyper-parameters and with hyper-parameter settings that we determined by our own on all datasets. For the HetRec and Amazon Instant Video datasets, all our baselines, to our surprise also including the TopPoular method, outperformed SpectralCF on all measures. However, when running the code on the provided MovieLens data splits, we found that SpectralCF was better than all our baselines by a huge margin. Recall@20 was, for example, 50% higher than our best baseline.

We therefore analyzed the published train-test split for the Movie-Lens dataset and observed that the popularity distribution of the items in the test set is very different from a distribution that would likely result from a random sampling procedure.[10] We then ran experiments with our own train-test splits also for the MovieLens dataset, using the splitting procedure described in the paper. We optimized the parameters for our data split to ensure a fair comparison. The results of the experiment are shown in Table 7. When using data splits that were created as described in the original paper, the results for the MovieLens dataset are in line with our own experiments for the other two datasets, i.e., SpectralCF in all configurations performed worse than our baseline methods and was outperformed even by the TopPopular method.

Table 7: Experimental results for SpectralCF (MovieLens1M, using own random splits and five repeated measurements).

| | Cutoff 20 | | Cutoff 60 | | Cutoff 100 | |
	REC	MAP	REC	MAP	REC	MAP
TopPopular	**0.1853**	**0.0576**	**0.3335**	**0.0659**	0.4244	**0.0696**
UserKNN CF	**0.2881**	**0.1106**	0.4780	**0.1238**	0.5790	**0.1290**
ItemKNN CF	0.2819	0.1059	0.4712	0.1190	0.5737	0.1243
$P^3\alpha$	0.2853	0.1051	0.4808	0.1195	0.5760	0.1248
$RP^3\beta$	**0.2910**	**0.1088**	**0.4882**	**0.1233**	**0.5884**	**0.1288**
SpectralCF	0.1843	0.0539	0.3274	0.0618	0.4254	0.0656

Figure 1 visualizes the data splitting problem. The blue data points show the normalized popularity values for each item in the training set, with the most popular item in the corresponding split having the value 1, ordered by decreasing popularity values. In case of random sampling of ratings, the orange points from the test set would mostly be very close to the corresponding blue ones. Here, however, we see that the popularity values of many items

---

[9]To assess the cold-start behavior, additional experiments are performed with fewer data points per user in the training set.
[10]We contacted the authors on this issue, but did not receive an explanation for this phenomenon.

in the test set differ largely. An analysis of the distributions with measures like the Gini index or Shannon entropy confirms that the dataset characteristics of the shared test set diverge largely from a random split. The Gini index of a true random split lies at around 0.79 for both the training and test split. While the Gini index for the provided training split is similar to ours, the Gini index of the provided test split is much higher (0.92), which means that the distribution has a much higher popularity bias than a random split.
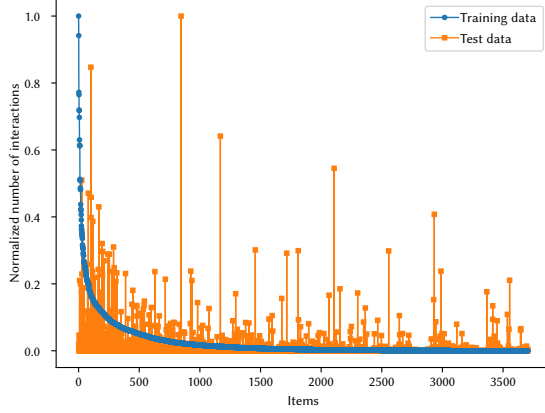

Figure 1: Popularity distributions of the provided training and test splits. In case of a random split, the normalized values should, on average, be close for both splits.

3.7 Variational Autoencoders for Collaborative Filtering (Mult-VAE)

Mult-VAE [24] is a collaborative filtering method for implicit feedback based on variational autoencoders. The work was presented at WWW '18. With Mult-VAE, the authors introduce a generative model with multinomial likelihood, propose a different regularization parameter for the learning objective, and use Bayesian inference for parameter estimation. They evaluate their method on three binarized datasets that originally contain movie ratings or song play counts. The baselines in the experiments include both a matrix factorization method from 2008 [18], a linear model from 2011 [33], and a more recent neural method [51]. Accoring to the reported experiments, the proposed method leads to accuracy results that are typically around 3% better than the best baseline in terms of Recall and the NDCG.

Using their code and datasets, we found that the proposed method indeed consistently outperforms our quite simple baseline techniques. The obtained accuracy results were between 10% and 20% better than our best baseline. Thus, with Mult-VAE, we found one example in the examined literature where a more complex method was better, by a large margin, than any of our baseline techniques in all configurations.

To validate that Mult-VAE is advantageous over the complex non-neural models, as reported in [24], we optimized the parameters for the weighted matrix factorization technique [18] and the linear model [33] (SLIM using Elastic Net) for the MovieLens and Netflix

datasets by ourselves. We made the following observations. For both datasets, we could reproduce the results and observe improvements over SLIM of up to 5% on the different measures reported in the original papers. Table 8 shows the outcomes for the Netflix datasets using the measurements and cutoffs from the original experiments after optimizing for NDCG@100 as in [24].

Table 8: Experimental results for Mult-VAE (Netflix data), using metrics and cutoffs reported in the original paper.

	REC@20	REC@50	NDCG@100
TopPop	0.0782	0.1643	0.1570
ItemKNN CF	0.2088	0.3386	0.3086
$P^3\alpha$	0.1977	0.3346	0.2967
$RP^3\beta$	0.2196	0.3560	0.3246
SLIM	0.2551	0.3995	0.3745
Mult-VAE	**0.2626**	**0.4138**	**0.3756**

The differences between Mult-VAE and SLIM in terms of the NDCG, the optimization goal, are quite small. In terms of the Recall, however, Mult-VAE improvements over SLIM seem solid. Since the choice of the used cutoffs (20 and 50 for Recall, and 100 for NDCG) is not very consistent in [24], we made additional measurements at different cutoff lengths. The results are provided in Table 9. They show that when using the NDCG as an optimization goal *and* as a performance measure, the differences between SLIM and Mult-VAE disappear on this dataset, and SLIM is actually sometimes slightly better. A similar phenomenon can be observed for the MovieLens dataset. In this particular case, therefore, the progress that is achieved through the neural approach is only partial and depends on the chosen evaluation measure.

Table 9: Experimental results for Mult-VAE using additional cutoff lengths for the Netflix dataset.

	NDCG@20	NDCG@50	REC@100	NDCG@100
SLIM	**0.2473**	**0.3196**	0.5289	0.3745
Mult-VAE	0.2448	0.3192	**0.5476**	**0.3756**

4 DISCUSSION

4.1 Reproducibility and Scalability

In some ways, establishing reproducibility in applied machine learning should be much easier than in other scientific disciplines and also other subfields of computer science. While many recommendation algorithms are not fully deterministic, e.g., because they use some form of random initialization of parameters, the variability of the obtained results when repeating the exact same experiment configuration several times is probably very low in most cases. Therefore, when researchers provide their code and the used data, everyone should be able to reproduce more or less the exact same results. Given that researchers today often rely on software that is publicly available or provided by academic institutions, the barriers regarding technological requirements are mostly low as well. In

particular, virtualization technology should make it easier for other researchers to repeat an experiment under very similar conditions.

Nonetheless, our work shows that the level of reproducibility is actually not high. The code of the core algorithms seems to be more often shared by researchers than in the past, probably also due to the fact that reproducibility has become an evaluation criterion for conferences. However, in many cases, the code that is used for hyper-parameter optimization, evaluation, data pre-processing, and for the baselines is not shared. This makes it difficult for others to validate the reported findings.

One orthogonal factor that can make reproducibility challenging is the computational complexity of many of the proposed methods. Ten years after the Netflix Prize and its 100 million rating dataset, researchers, in the year 2019, commonly use datasets containing only a few hundred thousand ratings. Even for such tiny datasets, which were considered unacceptably small a few years ago, hyper-parameter optimization can take days or weeks, even when researchers have access to GPU computing. Clearly, nearest-neighbor methods, as discussed in our paper, can also lead to scalability issues. However, with appropriate data pre-processing and data sampling mechanisms, scalability can also be ensured for such methods, both in academic and industrial environments [19, 26].

4.2 Progress Assessment

Despite their computational complexity, our analysis showed that several recently proposed neural methods do not even outperform conceptually or computationally simpler, sometimes long-known, algorithms. The level of progress that is achieved in the field of neural methods is, therefore, unclear, at least when considering the approaches discussed in our paper.

One main reason for this *phantom progress*, as our work shows, lies in the choice of the baselines and the lack of a proper optimization of the baselines. In the majority of the investigated cases, not enough information is given about the optimization of the considered baselines. Sometimes, we also found that mistakes were made with respect to data splitting and the implementation of certain evaluation measures and protocols.

Another interesting observation is that a number of recent papers use the neural collaborative filtering method (NCF) [14] as one of their state-of-the-art baselines. According to our analysis, this method is however outperformed by simple baselines on one dataset and does not lead to much better results on another, where it is also outperformed by a standard implementation of a linear regression method. Therefore, progress is often claimed by comparing a complex neural model against another neural model, which is, however, not necessarily a strong baseline. Similar observations can be made for the area of session-based recommendation, where a recent method based on recurrent neural networks [16] is considered a competitive baseline, even though almost trivial methods are in most cases better [29, 30].

Another aspect that makes it difficult to assess progress in the field lies in the variety of datasets, evaluation protocols, metrics, and baselines that are used by researchers. Regarding datasets, for example, we found over 20 public datasets that were used, plus several variants of the MovieLens and Yelp datasets. As a result, most datasets are only used in one or two papers. All sorts of metrics

are used (e.g., Precision, Recall, Mean Average Precision, NDCG, MRR etc.) as well as various evaluation procedures (e.g., random holdout 80/20, leave-last-out, leave-one-out, 100 negative items or 50 negative items for each positive). In most cases, however, these choices are not well justified beyond the fact that others used them before. In reality, the choice of the metric should depend on the application context. In some applications, for example, it might be important to have at least one relevant item at the top of the recommendations, which suggests the use of rank-based metrics like MRR. In other domains, high Recall might be more important when the goal is to show as many relevant items as possible to the user. Besides the unclear choice of the measure, often also the cutoff sizes for the measurement are not explained and range from top-3 or top-5 lists to several hundred elements.

These phenomena are, however, not tied to neural recommendation approaches, but can be found in algorithmic research in recommender systems also in pre-neural times. Considering the arguments from [27, 46], such developments are fueled by the strong focus of machine learning researchers on accuracy measures and the hunt for the "best" model. In our current research practice, it is often considered sufficient to show that a new method can outperform a set of existing algorithms on at least one or two public datasets on one or two established accuracy measures.[11] The choice of the evaluation measure and dataset however often seems arbitrary.

An example of such unclear research practice is the use of Movie-Lens rating datasets for the evaluation of algorithms for implicit feedback datasets. Such practices point to the underlying fundamental problem that research is not guided by any hypothesis or aim at the solution of a given problem. The hunt for better accuracy values dominates research activities in this area, even though it is not even clear if slightly higher accuracy values are relevant in terms of adding value for recommendation consumers or providers [20, 22, 52]. In fact, a number of research works exist that indicate that higher accuracy does not necessarily translate into better-received recommendations [4, 9, 13, 31, 37].

5 SUMMARY

In this work, we have analyzed a number of recent neural algorithms for top-n recommendation. Our analysis indicates that reproducing published research is still challenging. Furthermore, it turned out that most of the reviewed works can be outperformed at least on some datasets by conceptually and computationally simpler algorithms. Our work therefore calls for more rigor and better research practices with respect to the evaluation of algorithmic contributions in this area.

Our analyses so far are limited to papers published in certain conference series. In our ongoing and future work, we plan to extend our analysis to other publication outlets and other types of recommendation problems. Furthermore, we plan to consider more traditional algorithms as baselines, e.g., based on matrix factorization.

---

[11]From the 18 papers considered relevant for our study, there were at least two papers which proposed new DL architectures which were evaluated on a single private dataset and for which no source code was provided.

REFERENCES

[1] S. Antenucci, S. Boglio, E. Chioso, E. Dervishaj, K. Shuwen, T. Scarlatti, and M. Ferrari Dacrema. 2018. Artist-driven layering and user's behaviour impact on recommendations in a playlist continuation scenario. In *Proceedings of the ACM Recommender Systems Challenge 2018 (RecSys 2018)*. https://doi.org/10.1145/3267471.3267475 Source: https://github.com/MaurizioFD/spotify-recsys-challenge.

[2] 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 CIKM '09*. 601–610.

[3] Joeran Beel, Corinna Breitinger, Stefan Langer, Andreas Lommatzsch, and Bela Gipp. 2016. Towards reproducibility in recommender-systems research. *User Modeling and User-Adapted Interaction* 26, 1 (2016), 69–101.

[4] Jöran Beel and Stefan Langer. 2015. A Comparison of Offline Evaluations, Online Evaluations, and User Studies in the Context of Research-Paper Recommender Systems. In *Proceedings TPDL '15*. 153–168.

[5] Robert M Bell and Yehuda Koren. 2007. Improved neighborhood-based collaborative filtering. In *KDD cup and workshop at the KDD '07*. Citeseer, 7–14.

[6] Homanga Bharadhwaj, Homin Park, and Brian Y. Lim. 2018. RecGAN: Recurrent Generative Adversarial Networks for Recommendation Systems. In *Proceedings RecSys '18*. 372–376.

[7] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings SIGIR '17*. 335–344.

[8] Colin Cooper, Sang Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random walks in recommender systems: exact computation and simulations. In *Proceedings WWW '14*. 811–816.

[9] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. 2012. Investigating the Persuasion Potential of Recommender Systems from a Quality Perspective: An Empirical Study. *Transactions on Interactive Intelligent Systems* 2, 2 (2012), 1–41.

[10] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *Proceedings SIGIR '18*. 515–524.

[11] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings WWW '15*. 278–288.

[12] Association for Computing Machinery. 2016. Artifact Review and Badging. Available online at: https://www.acm.org/publications/policies/artifact-review-badging (Accessed March, 2018).

[13] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. 2014. Offline and Online Evaluation of News Recommender Systems at Swissinfo.Ch. In *Proceedings RecSys '14*. 169–176.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings WWW '17*. 173–182.

[15] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep Reinforcement Learning That Matters. In *Proceedings AAAI '18*. 3207–3214.

[16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *Proceedings ICLR '16*.

[17] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings KDD '18*. 1531–1540.

[18] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings ICDM '08*. 263–272.

[19] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation. In *Proceedings RecSys '17*. 306–310.

[20] Dietmar Jannach, Paul Resnick, Alexander Tuzhilin, and Markus Zanker. 2016. Recommender Systems - Beyond Matrix Completion. *Commun. ACM* 59, 11 (2016), 94–102.

[21] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *Proceedings RecSys '16*. 233–240.

[22] Joseph A. Konstan and John Riedl. 2012. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction* 22, 1 (2012), 101–123.

[23] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings KDD '17*. 305–314.

[24] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings WWW '18*. 689–698.

[25] Jimmy Lin. 2019. The Neural Hype and Comparisons Against Weak Baselines. *SIGIR Forum* 52, 2 (Jan. 2019), 40–51.

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

[27] Zachary C. Lipton and Jacob Steinhardt. 2018. Troubling Trends in Machine Learning Scholarship. arXiv:arXiv:1807.03341

[28] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*. Springer, 73–105.

[29] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of Session-based Recommendation Algorithms. *User-Modeling and User-Adapted Interaction* 28, 4–5 (2018), 331–390.

[30] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Performance Comparison of Neural and Non-Neural Approaches to Session-based Recommendation. In *Proceedings RecSys '19*. https://doi.org/10.1145/3298689.3347041

[31] Andrii Maksai, Florent Garcin, and Boi Faltings. 2015. Predicting Online Performance of News Recommender Systems Through Richer Evaluation Metrics. In *Proceedings RecSys '15*. 179–186.

[32] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2018. A Contextual Attention Recurrent Architecture for Context-Aware Venue Recommendation. In *Proceedings SIGIR '18*. 555–564.

[33] Xia Ning and George Karypis. 2011. SLIM: Sparse linear methods for top-n recommender systems. In *Proceedings ICDM '11*. 497–506.

[34] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. 2017. Updatable, Accurate, Diverse, and Scalable Recommendations for Interactive Applications. *ACM Transactions on Interactive Intelligent Systems* 7, 1 (2017), 1.

[35] Hans Ekkehard Plesser. 2017. Reproducibility vs. Replicability: A Brief History of a Confused Terminology. *Frontiers in Neuroinformatics* 11, 76 (2017).

[36] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *Comput. Surveys* 51, 4 (2018), 1–36.

[37] Marco Rossetti, Fabio Stella, and Markus Zanker. 2016. Contrasting Offline and Online Results when Evaluating Recommendation Algorithms. In *Proceedings RecSys '16*. 31–34.

[38] Noveen Sachdeva, Kartik Gupta, and Vikram Pudi. 2018. Attentive Neural Architecture Incorporating Song Features for Music Recommendation. In *Proceedings RecSys '18*. 417–421.

[39] Alan Said and Alejandro Bellogín. 2014. Rival: A Toolkit to Foster Reproducibility in Recommender System Evaluation. In *Proceedings RecSys '14*. 371–372.

[40] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings WWW '01*. 285–295.

[41] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent Knowledge Graph Embedding for Effective Recommendation. In *Proceedings RecSys '18*. 297–305.

[42] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings WWW '18*. 729–739.

[43] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. In *Proceedings SIGKDD '18*. 2309–2318.

[44] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D Convolutional Networks for Session-based Recommendation with Content Features. In *Proceedings RecSys '17*. 138–146.

[45] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *Proceedings RecSys '16*. 225–232.

[46] Kiri Wagstaff. 2012. Machine Learning that Matters. In *Proceedings ICML '12*. 529–536.

[47] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings KDD '11*. 448–456.

[48] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings KDD '15*. 1235–1244.

[49] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings SIGIR '06*. 501–508.

[50] Jun Wang, Stephen Robertson, Arjen P de Vries, and Marcel JT Reinders. 2008. Probabilistic relevance ranking for collaborative filtering. *Information Retrieval* 11, 6 (2008), 477–497.

[51] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings WSDM '16*. 153–162.

[52] Bo Xiao and Izak Benbasat. 2007. E-commerce Product Recommendation Agents: Use, Characteristics, and Impact. *MIS Quarterly* 31, 1 (March 2007), 137–209.

[53] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral Collaborative Filtering. In *Proceedings RecSys '18*. 311–319.