A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research

MAURIZIO FERRARI DACREMA, SIMONE BOGLIO, and PAOLO CREMONESI,
Politecnico di Milano, Italy
DIETMAR JANNACH, University of Klagenfurt, Austria

The design of algorithms that generate personalized ranked item lists is a central topic of research in the field of recommender systems. In the past few years, in particular, approaches based on deep learning (neural) techniques have become dominant in the literature. For all of them, substantial progress over the state-of-the-art is claimed. However, indications exist of certain problems in today's research practice, e.g., with respect to the choice and optimization of the baselines used for comparison, raising questions about the published claims. To obtain a better understanding of the actual progress, we have compared recent results in the area of neural recommendation approaches based on collaborative filtering against a consistent set of existing simple baselines. The worrying outcome of the analysis of these recent works—all were published at prestigious scientific conferences between 2015 and 2018—is that 11 of the 12 reproducible neural approaches can be outperformed by conceptually simple methods, e.g., based on the nearest-neighbor heuristic or linear models. None of the computationally complex neural methods was actually consistently better than already existing learning-based techniques, e.g., using matrix factorization or linear models. In our analysis, we discuss common issues in today's research practice, which, despite the many papers that are published on the topic, have apparently led the field to a certain level of stagnation.[1]

CCS Concepts: • **Information systems** → **Recommender systems**; *Collaborative filtering*; • **General and reference** → *Evaluation*;

Additional Key Words and Phrases: Recommender systems, deep learning, evaluation; reproducibility

ACM Reference format:
Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2020. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Trans. Inf. Syst.* 39, 2, Article 20 (January 2021), 49 pages.
https://doi.org/10.1145/3434185

1 INTRODUCTION

Personalized recommendations are a common feature of many modern online services, e.g., on e-commerce, media streaming, and social media sites. In many cases, these recommendations are

[1] This article significantly extends or own previous work presented in References [19, 23].

Authors' addresses: M. Ferrari Dacrema, S. Boglio, and P. Cremonesi, Politecnico di Milano, Milano, Italy; emails: maurizio.ferrari@polimi.it, simone.boglio@mail.polimi.it, paolo.cremonesi@polimi.it; D. Jannach, University of Klagenfurt, Klagenfurt, Austria; email: dietmar.jannach@aau.at.

1046-8188/2020/01-ART20 $15.00
https://doi.org/10.1145/3434185

generated using *collaborative filtering (CF)* techniques [7]. Such techniques leverage the preference or activity profiles of a large user community to predict which are the most relevant items for the individual customer. Early technical approaches to build collaborative filtering recommender systems (RS) were based on nearest-neighbor techniques and date back to the 1990s [41, 58]. In the following years numerous machine learning approaches were proposed for the *top-n* recommendation task, which is to rank the available items according to their assumed relevance for the user. In the most recent years, recommendation methods based on deep learning (DL) or "neural" technology have become particularly popular. While these methods are often computationally complex, their solid success in other application areas—such as language or image processing—led to deep learning techniques nowadays dominating the recommender systems research landscape.

However, there are a number of indications that using increasingly deeper learning methods is not as beneficial as one would expect. For example, in two recent papers [39, 78], the authors report that for a certain information retrieval task recent neural methods are actually not better than long-existing non-neural ones. In the domain of time series prediction, Makridakis et al. [46] compared various statistical methods with traditional and recent neural methods and observed that statistical methods were mostly favorable over even the most recent machine learning techniques. In the area of recommender systems, the empirical analyses in References [44, 45] showed that sometimes almost trivial methods can outperform the latest neural methods for the task of session-based recommendations. These findings are, however, not tied to recent deep learning approaches alone. The observation that the reported improvements in the literature for certain retrieval tasks "don't add up" was put forward back in 2009 [3]. Most often, the reason for such *phantom progress* lies in the choice and poor optimization of the baselines used in the experiments. Lin [40], while recognizing the progress made by transformer models for certain tasks of document retrieval, observed it is not rare to find articles comparing neural ranking algorithms only against weak baselines, raising questions regarding the empirical rigor in the field. For the domain of rating prediction for recommender systems, Rendle et al. [57] found that many algorithms that were published between 2015 and 2019 actually did not outperform longer-existing methods, when these were properly tuned.

In their paper, Rendle et al. attribute the problem not only to poorly optimized baselines but also to the sometimes missing standardized benchmarks. In some sense, this finding is surprising, since the research community over the last decades has indeed developed certain standards regarding the evaluation of *top-n* recommendation algorithms. Typically, the input to an algorithm is a matrix of user-item interactions and the task is to compute relevance scores for the missing entries or just to rank the items. A number of public datasets exists that can be used for reproducible evaluation, as well as various well-established evaluation metrics. Finally, the technical details of newly proposed algorithms are often laid out in great detail in scientific papers and sometimes authors even share the code they used in their experiments.

This methodological approach for benchmarking algorithms seems very solid at first sight and suitable to determine if an algorithm is favorable over another in a specific combination of *(i)* performance measure, *(ii)* evaluation procedure, and *(iii)* dataset, at least when we assume that both algorithms are properly optimized. However, the claims made in many research papers are much more general. As can be observed in other fields of computer science, many papers claim a significant improvement over the "state-of-the-art" but do not explicitly state such claim to be only supported under very specific experimental conditions. In today's recommender systems research scholarship the researcher has ample freedom in selecting the specific experimental conditions, i.e., which metrics, which protocol, which datasets, and which baselines to use.

Given these observations regarding potential methodological issues in our field, we were wondering to what extent recent deep learning methods actually help to achieve progress in creating *top-n* recommendation lists based on user-item rating matrices. To that purpose, we conducted an extensive set of experiments in which we tried to reproduce the results reported in papers that were recently published in top-tier scientific conferences. We scanned the proceedings of several conference series and identified 26 relevant papers. We could reproduce 12 (46%) of them with reasonable effort. After fine-tuning a number of established baselines algorithms, it turned out that in 8 of the reproducible cases, simple and long-known approaches (e.g., based on nearest neighbors) outperformed the latest neural methods. Furthermore, when including established linear models based on machine learning (e.g., based on matrix factorization), *we found that only 1 of 12 recent neural methods was clearly better than the non-neural baselines.* And even in this case, this was true for only one of the considered datasets. Overall, these results indicate certain signs of stagnation in the context of applying machine learning methods for *top-n* recommendation tasks, despite many papers published every year claiming substantial progress.

The rest of the article is organized as follows. Next, in Section 2, we describe how we selected relevant works considered in our study. Section 3 provides details about our research methodology, and Section 4 lists our results in detail. The potential reasons for the observed phenomena are finally discussed in Section 5.

2 IDENTIFYING RELEVANT AND REPRODUCIBLE WORKS

To make sure our research is neither focusing on a few hand-picked examples nor singling out any individual researcher, we followed a systematic approach to identify recent papers that are relevant for our analysis. To obtain a suitable sample for our research, we considered papers that fulfilled the following constraints.

(1) **Topic.** The paper proposed a new neural collaborative filtering recommendation method for *top-n* recommendation tasks. Hybrid techniques with a collaborative filtering component were considered as well. We limited ourselves to works that focus on the traditional item ranking task. Papers that dealt with other tasks, e.g., session-based recommendation or group recommendation, were not considered to be in the scope of our study. Furthermore, to be considered, a paper had to report at least one ranking or classification accuracy metric. Papers that focused solely on rating or relevance prediction tasks, e.g., using the RMSE, were not taken into account.

(2) **Publication date.** The paper was published between 2015 and 2018 in one of the following conferences: SIGIR, KDD, TheWebConf (WWW), IJCAI, WSDM, ACM RecSys. All of the mentioned conferences are typical outlets for recommender systems research in computer science. Furthermore, all of them, except ACM RecSys, are rated with A* in the Australian CORE ranking system. ACM RecSys, in contrast, is entirely focused on recommender systems, which is why we included it in the study as well. We identified relevant papers by scanning the proceedings of these conference series in a manual process.

(3) **Reproducible experimental setup.** The experimental setting was reproducible, with reasonable effort, based on source code published by the authors. This constitutes a precondition for the reproducibility of the published results.[2]
   - In particular, the source code should include the implementation of the model and its data structures, a train loop, and a way of computing the recommendations; see also the discussions by Collberg and Proebsting [15]. For some of the relevant papers we found

[2] In theory, research papers should contain all relevant information that are needed to implement the proposed algorithm. In reality, however, certain details are sometimes omitted in length-restricted conference papers. For that reason, and to ensure that the results that we report here are reliable, we followed a conservative approach and limited ourselves to the papers where the original authors themselves provided an implementation of their method.

Table 1. Statistics of Relevant and Reproducible Works on Deep Learning Algorithms for *Top-n* Recommendation per Conference Series from 2015 to 2018

Conference	Rep. Setup ratio	Reproducible Setup	Non-Reproducible Setup
KDD	3/4 (75%)	[32], [37], [73]	[67]
IJCAI	5/7 (71%)	[29], [80], [79], [13], [77]	[51], [76]
WWW	2/4 (50%)	[30], [38]	[66], [22]
SIGIR	1/3 (30%)	[21]	[48], [12]
RecSys	1/7 (14%)	[81]	[65], [8], [60], [68], [34], [71]
WSDM	0/1 (0%)		[75]
Total	12/26 (46%)		

that some source code was published but was missing major parts, e.g., it consisted only of a skeleton of the algorithm, or did not work. For those papers that were deemed relevant but for which the source code was not publicly available or was not runnable, we contacted all authors of the papers by e-mail. When there was no positive response after 30 days, we considered the source code of the paper to be not available.

- At least one of the datasets that were used in the original paper for evaluation was publicly available. In some cases, the authors also provided the data splits for training and testing that were used in their experiments. If the data splits were not available, then the experimental setup of the papers was considered as reproducible only if they contained sufficient information about data pre-processing and splitting.[3]

Following this approach, we identified **26** relevant papers. Of these, **12** were considered having a reproducible experimental setup, according to our classification scheme. Table 1 summarizes which works were considered relevant and which ones had a reproducible setup.[4]

Our first contribution in this work is therefore an analysis of the reproducibility—at least when using our specific practical definition—of research works published on neural collaborative filtering. We generally found that the share of papers that can be reproduced based on the provided source code by the authors is still relatively low. When looking at the statistics over the years, we can observe a certain trend towards authors sharing their source code more often. One possible reason is that reproducibility in general is considered a positive point in the reviewing process, e.g., at KDD.

3 EVALUATION METHODOLOGY

The core of our study was to re-run the experiments reported in the original papers following the original experimental methodology, including additional baseline methods which were systematically fine-tuned just like the newly proposed methods.

To ensure the reproducibility of this study, we share all the data, the source code used for pre-processing, hyperparameter optimization, algorithms, and the evaluation as well as all hyperparameter values and results online.[5]

[3] In case we encountered problems with the provided code, the data, or the reproduction of the results, we also contacted the authors for assistance.

[4] Among the papers with a reproducible experimental setup, there were some works where the numerical results could not be reproduced exactly, namely, References [77, 79, 80]. The reproducibility of the numerical results is however not a criterion for the inclusion or exclusion of the algorithm in our analysis. Our approach therefore combines aspects of reproducibility and replicability according to the terminology in the SIGIR guidelines: https://sigir.org/wp-content/uploads/2018/07/p004.pdf; see also ACM Artifact Review and Badging: https://www.acm.org/publications/policies/artifact-review-badging.

[5] https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation.

3.1 Measurement Approach

Our analysis of the relevant papers shows that researchers use all sorts of datasets, evaluation protocols, and metrics in their experiments, see also Section 5.2. To make our analyses and comparisons as fair as possible, we decided to run our evaluations in exactly the same way as the authors of the originally proposed method did, i.e., using their datasets, their protocol, and their performance metrics. To obtain a broader understanding of the model performance, we also included additional baselines.[6]

To ensure all algorithms are evaluated under the same conditions, we re-factored the original code so that we could include it in our evaluation framework along with the additional baselines. The core algorithm implementations remained unaltered. We evaluated the algorithms using the datasets reported in the original papers, provided that they were either publicly available or shared by the authors. We also used the original train/test split whenever the authors shared it; otherwise, we created the data split ourselves following the information provided in the original paper.

Extensive hyperparameter optimization was performed for all examined baselines. For the investigated neural methods, in all but one case, we relied on the optimal hyperparameters reported in the original papers. This is appropriate, as we used the same datasets and evaluation procedures in the experiments. The only exception is the SpectralCF algorithm (Section A.12), for which we performed a new hyperparameter optimization ourselves due to an issue with the provided dataset splits, as will be discussed later. Since the number of epochs and the stopping criteria are usually not described in the original papers, for all machine learning models we select the number of epochs via *early stopping*, see also Section 3.3.

The optimal hyperparameters were selected via a Bayesian search [2, 26, 31], using the Scikit-Optimize[7] implementation. We considered 50 cases for each algorithm during this search. The first 15 of them were used as initial random points. Once the optimal hyperparameters were determined, including the number of epochs, the final model was fitted on the union of train and validation data using those optimal hyperparameters. The considered hyperparameter ranges and distributions are listed in Appendix B.

3.2 Baselines

Over the last 25 years, a multitude of algorithms of different types were proposed. To obtain a picture that is as broad as possible, we selected algorithms of different families for inclusion in our measurements. An overview of all used baselines is given in Table 2 and the respective hyperparameter ranges are reported in Appendix B.

Our analysis shows that a large variety of other baselines is used in the works we investigated. This variety of baselines (together with the variety of datasets) represents one of the underlying problems that we identified, because it limits the comparability of results across papers. In our work, we therefore benchmark the proposed models against a consistent set of known algorithms of different families.

3.2.1 Popularity-based Ranking. Recommending the most popular items to everyone is a common strategy in practice. The method **TopPopular** implements this non-personalized recommendation approach. The popularity of an item is determined by its number of implicit or explicit ratings in the given dataset.

[6] An alternative would have been to integrate all methods in one unified framework for evaluation, as done in Reference [44], and evaluate them on a set of representative datasets. This approach would allow a direct comparison of neural approaches as in Reference [45], which was, however, not the goal of our work.

[7] https://scikit-optimize.github.io/.

Table 2. Overview of Baseline Methods

Family	*Method*	*Description*
Non-personalized	TopPopular	Recommends the most popular items to everyone [18]
Nearest-neighbor	UserKNN	User-based k-nearest neighbors [58]
	ItemKNN	Item-based k-nearest neighbors [61]
Graph-based	$P^3\alpha$	A graph-based method based on random walks [16]
	$RP^3\beta$	An extension of $P^3\alpha$ [54]
Content-based and Hybrid	ItemKNN-CBF	ItemKNN with content-based similarity [43]
	ItemKNN-CFCBF	A simple item-based hybrid CBF/CF approach [50]
	UserKNN-CBF	UserKNN with content-based similarity
	UserKNN-CFCBF	A simple user-based hybrid CBF/CF approach
Non-Neural Machine Learning	iALS	Matrix factorization for implicit feedback data [33]
	pureSVD	A basic matrix factorization method [18]
	SLIM	A scalable linear model [36, 52]
	$EASE^R$	A recent linear model, similar to auto-encoders [63]

3.2.2 Nearest-neighbor Methods. Nearest-neighbor techniques were used in the early GroupLens system [58] and first successful reports of collaborative filtering systems also used nearest-neighbor techniques [41]. We consider both *user-based* and *item-based* variants, **UserKNN** and **ItemKNN**.

Many variants of the basic nearest-neighbor prediction scheme were proposed over the years, see Reference [10] for an early performance comparison. In this work, we therefore consider different variations of the nearest-neighbor techniques as well. For both UserKNN and ItemKNN, the following hyperparameters can be set and were optimized in our experiments, their ranges are reported in Appendix B.

- *Neighborhood Size*: This main parameter determines how many neighbors are considered for prediction.
- *Similarity Measure*: We made experiments with the Jaccard coefficient [55] as well as Cosine [61], Asymmetric Cosine [1], Dice-Sørensen [20], and Tversky [69] similarities. Some of these similarity measures also have their own parameters, as reported in Appendix B, which we optimized as well.
- *Shrinkage*: As proposed in Reference [5], we used a parameter (the *shrink term*) to lower the similarity between items that have only few interactions in common. The shrinkage is applied to all similarities.
- *Feature Weighting*: Using feature weighting for ratings was proposed in Reference [74]. In our experiments, we both tested configurations with no weighting and weighting with either the TF-IDF or the BM25 scheme.
- *Normalization*: This setting determines if we should consider the denominator in the similarity measure as normalization. Only some of the similarity measures have this parameter.

3.2.3 Graph-based Methods. Traditional nearest-neighbor models consider "direct" neighborhoods by computing similarities between pairs of objects. Graph-based models can help to overcome this possible limitation relying on a broader interpretation of neighborhoods. In our study, we consider two such graph-based methods called $P^3\alpha$ [16] and $RP^3\beta$ [54]. Both methods often lead to good recommendation quality at low computational cost. Interestingly, based on the comparatively limited number of citations they received, these two methods appear to be almost unknown in the community and seldom used as baselines, despite the fact that they are very simple, effective and have been published in top-tier venues.

- ***$P^3\alpha$***: This method implements a two-steps random walk from users to items and vice-versa, where the probabilities to jump between users and items are computed from the normalized ratings raised to the power of α. The method is equivalent to a KNN item-based CF algorithm, with the similarity matrix being computed as the dot-product of the probability vectors [16]. In addition to what is described in the original algorithm, we normalize each row of the similarity matrix with its $l1$ norm. The hyperparameters of the algorithm include the size of the neighborhood and the value for α.
- ***$RP^3\beta$***: This is an improved version of $\mathrm{P}^3\alpha$ proposed in Reference [54]. In $\mathrm{RP}^3\beta$, each similarity between two items is computed with $\mathrm{P}^3\alpha$ and divided by the popularity of the items raised to the power of β. Again, we normalize each row of the similarity matrix with its $l1$ norm. If β is 0, then $\mathrm{RP}^3\beta$ is equivalent to $\mathrm{P}^3\alpha$. The hyperparameters of the algorithm are the size of the neighborhood and the values for α and β.

3.2.4 Content-based and Hybrid Methods. Some of the neural methods investigated in this article include side information about items or users. We have therefore included two simple baselines that make usage of content information.

- **ItemKNN-CBF**, **UserKNN-CBF**: A neighborhood-based content-based-filtering (CBF) approach, where we compute the item (or user) similarities based on the items' (or user's) content features (attributes) [43]. We tested the same set of similarity measures described for the collaborative KNN methods (Jaccard coefficient, Cosine, Asymmetric Cosine, Dice-Sørensen and Tversky similarity). The hyperparameters are the same as for the ItemKNN and UserKNN methods.
- **ItemKNN-CFCBF**, **UserKNN-CFCBF**: A hybrid algorithm based on item-item (or user-user) similarities and described in Reference [50]. The similarity between items is computed by first concatenating, for each item, the vector of implicit ratings (collaborative features) and the vector of item attributes (content features) and by later computing the similarity between the concatenated vectors. In case of user-user similarities the algorithm operates in a similar way, concatenating the vector of implicit ratings of each user with the user's content feature vector. The hyperparameters and similarity measures are the same as for ItemKNN, plus a parameter w that controls the relative importance of the content features with respect to the collaborative features. When w is 0, this algorithm is equivalent to the pure collaborative versions, either ItemKNN or UserKNN.

3.2.5 Non-neural Machine Learning Approaches. A wide variety of machine learning models were proposed for *top-n* recommendation tasks in the literature. In our experiments, we included a number of comparably basic models from the literature as representatives of which methods were often considered the state-of-the-art in pre-neural times.

- *Matrix Factorization (MF)* Techniques: The application of matrix decomposition methods for collaborative filtering problems was investigated already in the early years of recommender systems [9], and became a de-facto standard after the Netflix prize competition (2006–2009). We made experiments with many variants, but will limit our discussion to two main techniques that proved to consistently lead to competitive results among the different MF techniques.
  - **iALS**: In their seminal work [33], Hu et al. proposed an *Alternating Least Squares* approach for implicit feedback datasets, which turns implicit feedback signals into

confidence values. The authors also proposed a particular optimization method that has the advantage of scaling well on larger datasets. A number of hyperparameters can be tuned for the method, including the number of latent factors, the confidence scaling and the regularization factor.

  - **PureSVD**: This method corresponds to a basic matrix factorization approach as proposed in Reference [18]. To implement PureSVD, we used a standard SVD decomposition method provided in the `scikit-learn` package for Python.[8] The only hyperparameter of this method is the number of latent factors.

- *Sparse Linear Models (SLIM)*: SLIM was proposed as a well-performing regression-based method for *top-n* recommendation tasks in Reference [52]. In our work, we use the more scalable variant proposed in Reference [36] (**SLIM ElasticNet**), which learns the item similarity matrix one item at a time (e.g., one column w at a time) by solving a regression problem in such a way that the interactions for the target item y are learned by using all other interactions as training data. To implement *SLIM ElasticNet* we used a standard ElasticNet solver provided in the `scikit-learn` package for Python.[9] The hyperparameters of this method include the ratio of *l1* and *l2* regularizations as well as a regularization magnitude coefficient.
- ***EASE^R***: In a recent article [63] the author showed that an "embarrassingly shallow" linear model, which shares similarities with an auto-encoder, can produce highly accurate recommendations that often outperform existing and much more complex techniques. A peculiarity of this model is the existence of a closed-form solution for the training objective, which results in very fast training. The only hyperparameter is the choice of the regularization factor. This algorithm has been published in 2019 and, as such, the papers covered by our study could not include EASE^R as a baseline. However, we include EASE^R to investigate whether shallow auto-encoders are able to provide, on average, more accurate recommendations with respect to complex deep-learning architectures.

3.3 Early-stopping Approach

Many machine learning models are trained for a *number of epochs* in which the model's performance varies, first increasing and then stabilizing, while usually exhibiting some degree of variance. The number of epochs therefore represents another important parameter to be determined. However, it is worth noting that in the articles we have analyzed neither the number of epochs nor the stopping criteria are usually mentioned. The procedure in which this parameter was chosen in the original articles is therefore not clear. Looking at the code shared by the authors we could observe that, in some cases, the number of epochs was inappropriately selected via an evaluation done on the test data, therefore causing information leakage from the test data. In other cases, the reported metric values were inappropriately taken from different epochs.

Early stopping is a widely used technique to select the optimal number of train epochs and is available in many libraries like Keras.[10] The idea of early stopping is to periodically evaluate the model on the validation data, while the model is being trained, and stop the training when for a certain number of validation steps the model quality has not improved over the best solution

[8] https://scikit-learn.org/stable/modules/generated/sklearn.utils.extmath.randomized_svd.html.

[9] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html.

[10] For early stopping in Keras, see https://keras.io/callbacks/#earlystopping.

Table 3. Overview of Reproducible Papers

Paper	Conference
Collaborative Deep Learning for Recommender Systems (CDL) [73][11]	KDD '15
Collaborative Variational Autoencoder for Recommender Systems [37][12]	KDD '17
Neural Collaborative Filtering [30][13]	WWW '17
Deep Matrix Factorization Models for Recommender Systems [77][14]	IJCAI '17
Variational Autoencoders for Collaborative Filtering [38][15]	WWW '18
NeuRec: On Nonlinear Transformation for Personalized Ranking [80][16]	IJCAI '18
CoupledCF: Learning Explicit and Implicit User-item Couplings in Recommendation for Deep Collaborative Filtering [79][17]	IJCAI '18
DELF: A Dual-Embedding based Deep Latent Factor Model for Recommendation [13][18]	IJCAI '18
Outer Product-based Neural Collaborative Filtering [29][19]	IJCAI '18
Leveraging Meta-path based Context for *top-n* Recommendation with a Neural Co-attention Model [32][20]	KDD '18
Collaborative Memory Network for Recommendation Systems [21][21]	SIGIR '18
Spectral Collaborative Filtering [81][22]	RecSys '18

found so far. Early stopping has the advantage of selecting the number of epochs with a transparent criterion, avoiding arbitrary manual optimization, and often results in shorter training times.

To implement early stopping, we use two independent copies of the current model. One is the model that is still being trained, the other is the model frozen at the epoch with the best recommendation quality found so far. If the trained model, after further epochs, exhibits better recommendation quality than the best one found so far, then the best model is updated. Since the evaluation step is time consuming, we run five train epochs before each validation step. Moreover, we choose to stop the training if for five consecutive validation steps the recommendation quality of the current model is worse than the best one found so far.

4 RESULTS—ANALYSIS OF REPRODUCIBILITY AND PROGRESS

In this section, we summarize the main observations of our experiments. For each analyzed paper, we describe the basic idea of the method and the summary of our results. A more detailed analysis is reported in Appendix A, where we describe *(i)* the baseline algorithms and datasets that were used in the original paper; *(ii)* the outcomes reported in the original work; *(iii)* our results after including and optimizing additional baselines.

[11] https://github.com/js05212/CDL.
[12] https://github.com/eelxpeng/CollaborativeVAE.
[13] https://github.com/hexiangnan/neural_collaborative_filtering.
[14] The source code was not publicly available but the authors shared it with us upon request.
[15] https://github.com/dawenl/vae_cf.
[16] https://github.com/cheungdaven/NeuRec.
[17] https://github.com/zhqgui/CoupledCF.
[18] The source code was not publicly available but the authors shared it with us upon request.
[19] https://github.com/duxy-me/ConvNCF.
[20] https://github.com/librahu/MCRec.
[21] https://github.com/tebesu/CollaborativeMemoryNetwork.
[22] https://github.com/lzheng21/SpectralCF.

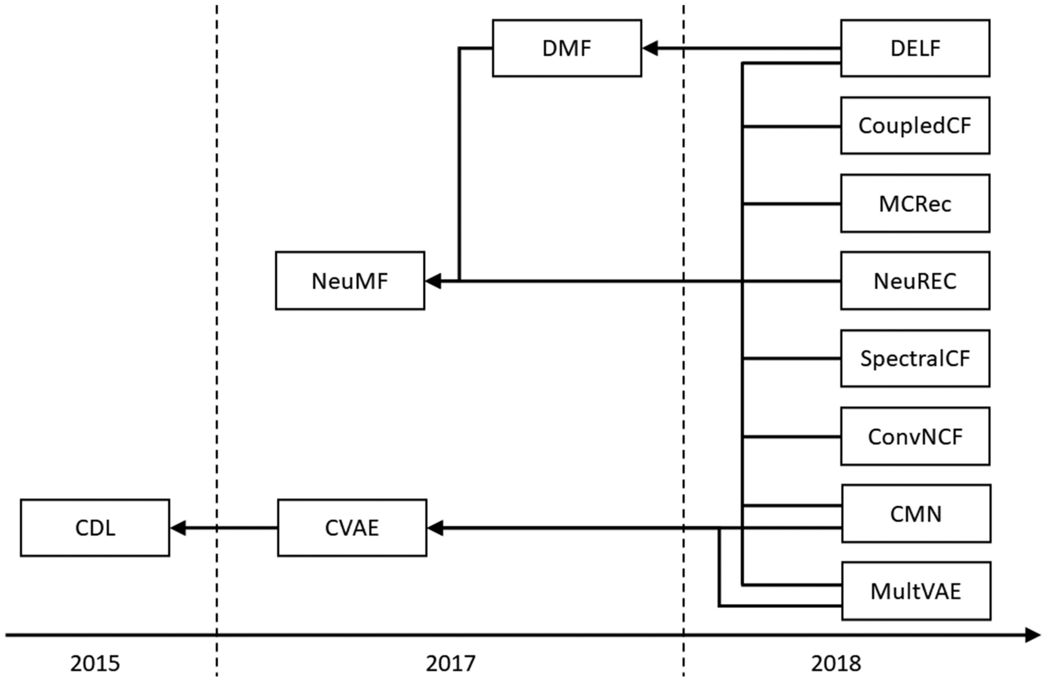


Fig. 1. Overview of Neural Methods. Arrows indicate when a newer method used another one as baseline in the experiments.

The analyzed papers were published between 2015 and 2018. We organize the discussion of the papers mostly by year of publication. The list of considered papers is shown in Table 3. An overview of the temporal development and the dependencies between the approaches can be found in Figure 1.

The experimental evaluation reported in this article required significant computational effort. In total, we report the recommendation quality of more than 900 trained models (counting both baselines and reproducible deep learning algorithms). When taking into account the hyperparameter tuning procedure, 41,000 models were fitted, corresponding to a total computation time of 253 days.[23]

4.1 Collaborative Deep Learning for Recommender Systems (CDL)

Method. CDL is the earliest method in our analysis, published at KDD '15 [73].[24] CDL is a hybrid method that applies deep learning to jointly learn a deep representation of content information and collaborative information. Technically, it is a probabilistic feed-forward model for joint learning of a stacked denoising autoencoder and collaborative filtering.

Result summary. According to our experiments, simple hybrid baselines outperform CDL in three of four datasets. On a dense dataset, CDL is also outperformed by pure collaborative

[23] The computation time refers to the total instance time for one AWS instance p3.2xlarge, with 8 vCPU, 30GB RAM, and one Tesla V100-SXM2-16GB GPU. The detailed measurements are available in the online material.
[24] https://github.com/js05212/CDL.

baselines with recommendation lists shorter than 100 items. CDL is only better than our baselines on one small and very sparse dataset with only one interaction per user in the training set. From a methodological perspective, there is no indication of why comparably long list lengths, up to 300, were used for evaluation in the paper and why no measurements were reported for list lengths below 50, which is commonly the case in the literature.

4.2 Collaborative Variational Autoencoder (CVAE)

Method. Like CDL, the *CVAE* method [37][25] is a hybrid technique that relies both on content information and collaborative features for recommending. The work was published at KDD '17. Technically, the model learns deep latent representations from content data in an unsupervised manner and also considers implicit relationships between items and users from both content and ratings. Unlike previous works proposing denoising autoencoders, CVAE learns a latent distribution for content in the latent space instead of the observation space through an inference network.

Result summary. While CVAE, according to the original experiments, outperforms previous neural methods including CDL [73], our experiments indicate these neural methods were not necessarily strong baselines. Our own experiments confirm the improvements of CVAE over the previous CDL method. However, similarly to CDL, simple hybrid methods outperformed CVAE in three of four datasets. CVAE is instead superior in one very sparse dataset that had only one interaction per user in the training set.

4.3 Neural Collaborative Filtering (NCF)

Method. The *Neural network-based Collaborative Filtering (NCF)* [30][26] framework was presented at WWW '17 and rapidly became very influential, being used as a baseline for most later neural recommendation approaches, as shown in Figure 1. The framework generalizes matrix factorization in a way that the commonly used inner product is replaced by a neural architecture, which can learn different types of functions from the data and therefore can also model non-linearities. Different variants are considered in the paper: Generalized Matrix Factorization, Multi-Layer Perceptron, and Neural Matrix Factorization, where the last one, called NeuMF is an ensemble of the other two. In our evaluation, we only consider Neural Matrix Factorization, because this method led to the best results.

Result summary. In our experiments, we could observe that, for one of two datasets, nearest-neighbor methods outperform NCF on all measurements. For the other dataset, NFC was better than the nearest-neighbor techniques, but not better than non-neural machine learning methods. Regarding methodological issues, we observed in the original source code that the number of training epochs, which should be optimized on the validation set, was erroneously optimized on the test data.[27]

4.4 Deep Matrix Factorization (DMF)

Method. Deep Matrix Factorization Models (DMF) were proposed at IJCAI '17 [77].[28] As an input to their model, the authors first build a user-item matrix from explicit ratings and implicit feedback, which is then used by a deep *structure learning* architecture. One key aspect here is that a common low-dimensional space for representing users and items is used. Furthermore, the authors develop

[25] https://github.com/eelxpeng/CollaborativeVAE.

[26] https://github.com/hexiangnan/neural_collaborative_filtering.

[27] In addition, Rendle et al. [56] confirmed that NCF is not able to consistently outperform simple non-neural baselines and experimentally showed that learning a dot product with a multilayer perceptron is not trivial.

[28] The source code was not publicly available but the authors shared it with us upon request.

a new loss function based on cross entropy that considers both implicit feedback and explicit ratings.

Result summary. The proposed method is outperformed on three of four datasets by our long-known baselines. Only for one very sparse dataset, the proposed method was better than our baseline methods, in particular with respect to the Hit Rate.[29] Regarding methodological aspects, we observed in the source code that the Hit Rate and NDCG results are reported as the best value obtained by evaluating on the test data during training, regardless of the epoch. In our experiments, we report the results associated to the epoch selected via early stopping on the validation data.

4.5 Variational Autoencoders for Collaborative Filtering (Mult-VAE)

Method. In Reference [38],[30] the authors propose a collaborative filtering method for recommendation based on implicit feedback using variational autoencoders. The method is called *Mult-VAE* and was presented at WWW '18. Technically, the paper introduces a generative model with multinomial likelihood and a different regularization parameter for the learning objective, and uses Bayesian inference for parameter estimation. The authors furthermore show that there is an efficient way to tune the parameter using annealing.

Result summary. We could reproduce the results reported in the original paper for two of three datasets and found that the proposed method outperforms all neighborhood-based baselines on all metrics by a large margin. The algorithm also outperforms all non-neural machine-learning baselines on one dataset. For the second dataset, however, the SLIM method was better when the optimization target was the same as the evaluation measure (NDCG).

4.6 NeuRec: On Nonlinear Transformation for Personalized Ranking

Method. NeuRec [80][31] was presented at IJCAI '18. The work aims at learning user-item relationships from implicit feedback and combines latent factor models with neural networks to capture both linear and non-linear dependencies in the data. Technically, the user-item interaction matrix is first mapped into a low-dimensional space with multi-layered networks. Recommendations are then generated by computing the inner product of item and user latent factors. A user-based and an item-based variant are proposed.

Result summary. Even though the authors published a runnable implementation of their method and provided detailed information on the hyperparameters, we could not obtain the results reported in the original paper. We contacted the authors but even with their help the reason for this differences could not be clarified. In our experiments, the method is consistently outperformed by many of our baselines on three of four commonly used datasets. Only on one small dataset and for one individual measure at a short list length the proposed method is slightly better than our baselines. Regarding methodological aspects, we found again that the number of epochs was erroneously optimized on the test data, furthermore based on the provided source code, the best results of NeuRec are reported for different measures at potentially different training epochs. In our experiments, we report the results associated to the epoch selected via early stopping on the validation data.

[29] The Hit Rate measures if a single hidden true positive was part of the top-n recommendation list, see, e.g., Reference [79] for a formal definition. The metric is similar to Recall.

[30] https://github.com/dawenl/vae_cf.

[31] https://github.com/cheungdaven/NeuRec.

4.7 CoupledCF: Learning Explicit and Implicit User-item Couplings

Method. CoupledCF [79][32] was also presented at IJCAI '18. The approach is based on the observation that users and items in real-world datasets are not independent and identically distributed. The proposed method therefore aims to learn implicit and explicit couplings between users and items and to thereby leverage available side information (e.g., user demographics, item features) more effectively. Technically, a complex architecture is used, involving a CNN for learning the couplings based on the side information and a deep CF model that considers explicit and implicit interactions between users and items.[33]

Result summary. For one of two datasets, the non-deep machine learning models were consistently better than the proposed method. For the second dataset, nearest-neighbor methods were preferable. Regarding methodological aspects, from the available source code, we could observe that the number of training epochs was selected using the test data and that the provided data splits exhibits inconsistencies with the splitting procedure described in the paper. Furthermore, as described in the original paper, the default hyperparameters for some baselines are used, meaning they were not properly optimized.

4.8 DELF: A Dual-embedding-based Deep Latent Factor Model for Recommendation

Method. The *DELF* model [13],[34] presented at IJCAI '18, was designed for *top-n* recommendation tasks given implicit feedback data. Inspired by previous work (NSVD) [53], the authors propose to learn *dual* embeddings to capture certain interactions in the data. Instead of using only the common user embedding, the authors propose to learn an additional item-based user embedding and vice versa for item embeddings. The embeddings are then combined to model non-linear interactions between users and items within a deep learning architecture. Through this approach the authors generalize ideas of NSVD and Neural Collaborative Filtering (NCF). Two variants of the approach, *DELF-MLP* and *DELF-EF* were investigated in the original paper.

Result summary. Two datasets, *MovieLens* and *Amazon Music*, were used for the evaluation. On the movies dataset, *DELF* was consistently outperformed by our machine learning baselines. The performance of *DELF* on the music dataset was, however, substantially better than all baselines. An investigation of this phenomenon revealed that when using the *Amazon Music* with a time-based splitting criterion, more than half of the test items never appeared in the training set, which is an uncommon setup for evaluating pure collaborative filtering approaches. The relatively high accuracy of *DELF* therefore stems from its tendency to push cold items—for which it could not have learned anything in the training phase—to the top end of the recommendation lists. An additional experiment in which cold items are not considered in the evaluation—which is a more suitable setup when evaluating pure collaborative filtering methods—shows that the performance of *DELF* again drops below that of the machine learning baselines. Looking at methodological aspects, like in other works discussed here, the authors did not optimize the number of epochs on the validation set but took the best values obtained when testing.

[32] https://github.com/zhqgui/CoupledCF.

[33] The claim that the outer product of embeddings is equivalent to an image and that the CNN allows to model embedding correlations has been later questioned by Ferrari Dacrema et al. [24]. Similar considerations apply for ConvNCF, discussed in Section 4.9.

[34] The source code was not publicly available but the authors shared it with us upon request.

4.9 Outer Product-based Neural Collaborative Filtering (ConvNCF)

Method. The *ConvNCF* method [29][35] was presented at IJCAI '18. Its main idea is to explicitly model the pairwise correlations between the dimensions of the embedding using an outer product. With this technique, the authors aim to create an *interaction map*, which is more expressive than existing methods that use simple concatenations of embeddings or element-wise products.[33]

Result summary. Two datasets were used for the evaluation, traditional nearest-neighbor baselines are consistently better than the proposed method on one dataset and better except for one measurement on the other. Regarding methodological aspects, based on the provided source code the number of epochs, as in other papers, was determined on the test data. Furthermore, the authors decided to set the embedding size to the constant value of 64 for all baselines. However, the embedding size is a hyperparameter to be tuned for each dataset and for each embedding-based algorithm, as different models with different objective functions or training procedures will likely require different values for it.

4.10 Leveraging Meta-path-based Context (MCRec)

Method. The *MCRec* [32][36] method was published at KDD '18. It is a hybrid method that uses side information about the recommendable items in the recommendation process. The side information is represented as a network structure, and meta-paths are relation sequences that connect objects in this graph. Technically, the authors use a priority-based sampling technique to select more informative paths instances and a novel co-attention mechanism to improve the representations of meta-path-based context, users and items.

Result summary. The authors provided an implementation that had the meta-paths hard-coded, and we therefore could reliably reproduce the results only for the small MovieLens dataset. For this dataset, however, it turned out that the traditional item-based nearest-neighbor technique was better than MCRec on all performance measures.

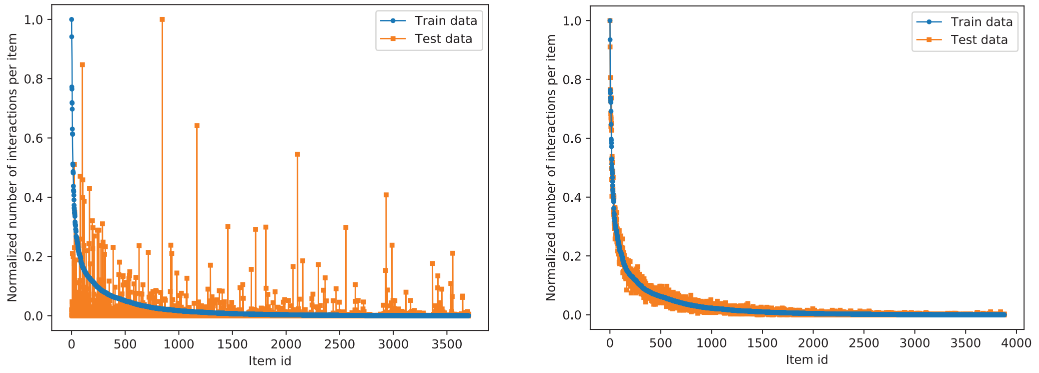
In the context of this work, additional problems were identified based on the provided source code. For example, the accuracy metrics reported by the source code correspond to the maximum values that are obtained across different epochs when evaluating on the test data. Furthermore, in the original article, the hyperparameters of the examined baselines were said to be taken from the original papers and not optimized for the datasets used in the evaluation. Finally, the NDCG metric was implemented in an uncommon way.

4.11 Collaborative Memory Network for Recommendation System (CMN)

Method. Collaborative Memory Networks (CMN) [21][37] was presented at SIGIR '18, and it represents a collaborative-filtering approach based on memory networks and neural attention mechanisms. The underlying idea of the approach is to combine latent factor with neighborhood (memory-based) approaches in a non-linear fashion.

Result summary. For two datasets of three, the recent CMN method was outperformed by most of the simpler baselines. On the third dataset, CMN was much better than our personalized baselines, but this dataset has such a skewed distribution that recommending the most popular items to everyone was by far the most effective method in this evaluation. The relatively good performance of CMN in this setting is therefore attributed to the higher popularity bias of CMN.

[35] https://github.com/duxy-me/ConvNCF.
[36] https://github.com/librahu/MCRec.
[37] https://github.com/tebesu/CollaborativeMemoryNetwork.



(a) Normalized popularity distributions of the train and test splits provided by the original authors.

(b) Normalized popularity distributions of the train and test splits generated by us.

Fig. 2. Normalized popularity distributions of the train and test splits for SpectralCF, the value 1 corresponds to the most popular item in that split. For a random split, as can be seen in panel (b), the normalized values of both splits are, on average, similar. In the split provided by the original authors, however, as can be seen in panel (a), train and test data have quite different distributions.

4.12 Spectral Collaborative Filtering (SpectralCF)

Method. SpectralCF, presented at RecSys '18 [81],[38] is a graph-based approach. Users and items are represented as a bipartite graph. The novelty of this method is a convolution approach that operates on the *spectral domain*. The method considers both proximity and connectivity information in the graph, which is assumed to be particularly helpful for cold-start problems.

Result summary. In our initial experiments, we found that the algorithm was competitive with our baselines only for one of three datasets. Specifically, it was the dataset for which the authors shared the train-test split. An investigation revealed the distribution of the data in the provided test set was very different from what we would likely obtain by applying a random sampling procedure. After creating the train-test splits by our own, we found that SpectralCF does not work as expected and consistently exhibits lower performance when compared with personalized and even non-personalized baselines.

To illustrate the data splitting problem, we compared the popularity distribution of the items in the training and the test split in the provided data; see Figure 2(a). The figure plots the normalized popularity (i.e., the popularity of an item over the popularity of the most popular item) for both training and test items. Items are ranked based on their popularity in the training set, with the most popular training items being on the left. In case of a true random split, the normalized popularity values of the items in the training and the test split should be relatively close. However, the figure shows the split provided by the authors has some major deviations. Figure 2(b) shows the popularity distributions of our random split, which are almost identical between training and test sets.

5 DISCUSSION

Our work indicates that rather limited progress was made through deep learning approaches applied to the classic *top-n* recommendation problem, at least when considering the papers that were analyzed in the present work and that were published at top-level scientific conferences. While

[38] https://github.com/lzheng21/SpectralCF.

many papers claiming to make advances over the state of the art were published, these mostly seemed to amount to phantom progress. In this section, we will review the possible causes for the apparent stagnation in this field. We hope that an increased awareness of these causes can help ensuring progress in the future and lead to the development of even more powerful top-n recommendation approaches. We will structure our discussions along three dimensions: reproducibility, methodology, and fundamental considerations. Please note that almost all discussed issues are not tied to deep learning techniques, and our article should therefore not be understood as a criticism of deep learning in general or in recommender systems research.

5.1 Reproducibility

The reproducibility statistics for the conferences we analyzed in this study are reported in Table 1. Using our specific criteria, we found that 12 of 26 papers could be reproduced based on the provided code and publicly available data sets. In recent years, we could observe an increased trend for researchers to share the source code of their algorithms and experiments, probably due to the increased importance of reproducibility as a criterion in the academic peer-reviewing process. Nonetheless, we found that for more than half of the papers the source code was either not provided or it was lacking important details to the extent that a reliable reproduction of the results was made difficult or impossible.

The reasons for not sharing the code, not even in private with other researchers, sometimes remain obscure. Technically, a scientific paper should contain all necessary information for others to reproduce the work. Nonetheless, providing the artifacts (code, data sets) that were used in the experiments is important, as details may sometimes be missing from the papers, e.g., for space reasons, that could have a significant impact on the observed results. Generally, however, establishing reproducibility in the context of algorithmic proposals, as discussed in our article, is relatively easy compared to other disciplines. Nowadays, applied research in deep learning is mostly based on public libraries and freely accessible software tools. Virtualization technology can also be leveraged effectively to make the entire software environment used for the experiments available to the community. Furthermore, while many algorithms have some non-deterministic elements, the overall performance is usually quite stable over several runs, which should allow other researchers to obtain at least comparable results when using the artifacts of the original authors.

Sometimes, researchers argue that they cannot provide the code because of the rules of the organization employing them. In some cases, the data used in the experiments is also not made available to the community, which means that no one will ever have the chance to verify, and possibly contest or falsify, the claimed results under the exact same conditions. This puts academic researchers in a certain dilemma. The scientific community is highly interested in both real-world problems and solutions that work in practice. From a scientific perspective, however, reproducibility and verifiability (or falsifiability) are the main guiding principles of scientific research.

In the context of our study, we contacted the authors of papers where the artifacts were not publicly available. While some of the authors responded to our inquiries and shared their code with us, we found that in the great majority of cases (10 of 14 for non-reproducible papers), we received no reply. We can only speculate about the reasons for these phenomena. Clearly, providing code for others can lead to a substantial amount of extra work for the researcher. Furthermore, if reviewers do not put much emphasis on this, there might be not enough incentives for the researcher to go the extra mile [15, 64]. At the same time, publishing all artifacts can make certain assumptions or limitations of a method more obvious, e.g., when the scalability of a method is limited. In general, however, it should be in the interest of the community, and hence of researchers themselves, that others can reproduce, confirm or falsify their results, as this is a cornerstone of scientific research in the first place. For a discussion of the role of reproducibility as a scientific principle in other areas of computer science, see, e.g., Reference [25].

5.2 Methodological Issues

Throughout the article, we reported a number of methodological issues that contributed to the limited progress we have observed in the analyzed articles. Here, we summarize our main observations.

- *Choice of baselines:* Algorithms for *top-n* recommendation problems have been investigated for decades, and now it is less than clear what represents the state-of-the-art. Determining the state-of-the-art can be difficult or even impossible due to the fact that there exist no general "best" algorithm. In fact, performance rankings of algorithms depend on a variety of factors, including the dataset characteristics or the evaluation procedure and measure. Another observation is that often only complex machine learning methods are considered as baselines. Comparably simple methods like $P^3\alpha$ and $RP^3\beta$ are not widely known in the community, even though they were published at top-level venues and often lead to strong results.
- *Propagation of weak baselines:* Nowadays, methods like NCF are often considered as competitive state-of-the-art baselines, even though our analysis showed that they are often not better than relatively simple and well-known techniques.
- *Lack of proper tuning of baselines:* This is probably the most striking observation of our analysis and is not specifically tied to deep learning approaches [57] or to recommendation problems [39, 40]. Researchers apparently invest significant efforts in optimizing their own new method but do not pay the same attention to their baselines. Sometimes, authors simply pick the hyperparameter settings reported to be optimal from a previous paper, even though those may refer to a different dataset or experimental procedure. Probably, this behavior might be the result of a *confirmation bias*, i.e., the tendency to search for results that affirm rather than refute prior research hypotheses.

Regarding the choice of the baselines, we found that in some cases researchers reuse the experimental design that was used in previous studies, i.e., they use the same datasets, evaluation protocol, and metrics, to demonstrate progress. This is in principle very meaningful as it helps the comparison of results. However, this also means that the experimental setup is not questioned anymore, because it has been used by many researchers before. Some papers in our present study are based on two quite small CiteULike datasets that were used in some early deep learning papers. These datasets were reused by other authors later on, without questioning if these quite small datasets were representative for a larger set of real-world recommendation problems or whether they were useful at all to analyze certain phenomena. In our analysis, we even found that the Epinions dataset had such characteristics that the non-personalized recommendation of the most popular items was favorable over any personalized technique.

In general, the reuse of experimental designs is still the exception rather than the norm. In our analysis, we found that researchers use a large variety of evaluation protocols, datasets, and performance measures in their experiments. This is expected in articles having different goals and aimed at different scenarios. However, in most cases, there is no particular argumentation on why a certain metric is used or why particular datasets from a certain domain serve as a basis for the research. To illustrate these phenomena, we show in Table 4 which datasets were used for evaluation in the reproducible papers.

Besides the fact that 12 reproducible articles used 18 different datasets, the authors also relied on quite a number of different data splitting procedures, including *leave-one-out*, *leave-last-out*, *80/20*

Table 4. Datasets Used by Reproducible Papers

Dataset	Paper	Dataset	Paper
Amazon Movie	[77]	Amazon Music	[77], [13]
CiteULike-a	[21], [37], [73]	CiteULike-t	[37], [73]
Epinions	[21]	FilmTrust	[80]
Frappe	[80]	Gowalla	[29]
LastFM	[81], [32]	Movielens Hetrec	[80]
Movielens100K	[32], [77]	MovieLens1M	[81], [30], [80], [79], [13]
MovieLens20M	[38]	MSD	[38]
Netflix Prize	[73], [38]	Pinterest	[21], [30]
Tafeng	[79]	Yelp	[81], [32], [29]

training/test split, *hold-out of users*, or *retain only 1 or 10 interactions per user*. When evaluating, researchers somewhat arbitrarily used 50, 100, or 1000 negative samples per positive sample for ranking. Moreover, the metrics–including the Hit Rate, Precision, Recall, MAP, or NDCG—were measured on a variety of cut-off thresholds between 1 and 300. In addition, several pre-processing strategies were applied, for example, retaining only users or items with an arbitrary minimum number of interactions. In most cases, no justification is provided for why a particular choice of pre-processing, metrics and cut-off was selected and which relevant scenario aims to represent. As a result—as recently pointed out by Reference [11]—each of these choices can lead to comparisons that are vulnerable to misinterpretation and that may lead to different or even opposite outcomes, depending on the exact combination of the used settings.

One possible way to obtain a better understanding regarding what represents the state-of-the-art could be the adoption of standardized evaluation setups, where all regarded algorithms are evaluated in the same experimental conditions. Such an approach was for example followed in Reference [44] for session-based recommendation problems, where the same set of measurements was applied for various datasets. In pre-neural times, such comparisons were often enabled by the use of recommender systems libraries like MyMediaLite[39] or LibRec,[40] which also included evaluation code. In contrast, the works analyzed in our study mostly implemented the evaluation routines from scratch, leading to certain problems as discussed in the article.

Regarding the issue of baselines often not being properly tuned, one problematic factor with deep learning methods is that most of them are computationally complex, and some of them have a large number of hyperparameters to tune. Furthermore, it is usually also possible to vary the structure of the network in terms of the layers and nodes in each layer. An exhaustive search through the hyperparameter space for a large number of network architectures is therefore often not possible in reasonable time.

Regarding absolute running times, let us consider a few examples.[41]

- The often-cited NCF (NeuMF) method (Section A.3) needs four hours to train on one of our machines for the popular MovieLens1M dataset. Training the best-performing SLIM method on the same machine and dataset requires only 4 minutes. Interestingly, the more

[39] http://www.mymedialite.net/index.html.

[40] https://www.librec.net/.

[41] All reported measurements were made on the same hardware: one AWS instance p3.2xlarge, with 8 vCPU, 30 GB RAM, and one Tesla V100-SXM2-16 GB GPU. All deep-learning algorithms (with the exception of CDL) were trained using the GPU. All of our baselines were trained only on the CPUs. The detailed measurements are available in the online material (Section 3).

recent EASE^R method has better accuracy than NeuMF, but only needs about 12 s for training on the same machine. Training NeuMF on the slightly larger Pinterest dataset (1.5M interactions) took more than two days, which is again much higher when compared to the training times of SLIM (12 min) or EASE^R (2 min).

- Similar observations can be made for the early CDL method (Section A.1). On the larger CiteULike-a dataset with about 55k interactions, CDL needs almost two hours for training, the EASE^R method 2 min, and our well-performing ItemKNN CFCBF hybrid less than 10 seconds to pre-compute the neighborhoods.
- The Mult-VAE method (Section A.2), which significantly outperformed our baselines on one dataset, seems to be also favorable over both baselines and other neural methods in terms of training times. For the relatively large MovieLens20M dataset, training needs about 21 min on our machine. This is much faster than the SLIM algorithm, which needs almost two hours and was competitive with Mult-VAE on the other dataset. EASE^R, which is on par or better than SLIM, is again favorable here, requiring only about 3 min for training.
- The algorithm with the longest training time on our machine is DMF. Its best variant, based on binary cross entropy, requires almost 5 days of training on the MovieLens1M dataset, while the simple iALS baseline, which outperforms DMF on all measures, requires only *4 min* of training.

Limited scalability is not a methodological issue per se, and we do not argue that scalability necessarily has to be a main focus of academic researchers, in particular when it comes to new technical ideas and proposals. Once the ideas are assumed to be viable and effective, questions related to the practical deployment can be addressed in subsequent research, e.g., by using more shallow models that implement the same concepts.

However, today's often enormous costs for tuning the baselines may lure researchers into taking hyperparameter settings from previous papers—even though they were determined for different evaluation setups—and reuse them in their own experiments. Another possible consequence of the sometimes high computation costs can be that researchers often do not apply cross-validation procedures. Instead, they rather base their conclusions on one single train-test split of the data. For this reason, in many research works the statistical significance of the observed outcomes is not reported.

Besides the general issues reported so far, we observed a number of further more technical issues. These include uncommon implementations of ranking measures, non-randomized data splits, reporting best results across different epochs, and determining the best number of epochs on the test set.

5.3 Fundamental Issues

Among the factors that may contribute to the apparent stagnation observed both in this study and in other fields [3, 46] are today's incentive mechanisms in the academic system and our established research practices; see also Reference [42]. Regarding the incentive system, researchers are more and more evaluated based on the citations their works receive. This might lead to the phenomenon that researchers develop a tendency to investigate problems that are popular and (easily) "publishable" [72]. With the current strong interest in neural algorithms [39], papers that do not propose or use deep learning approaches might get criticized by reviewers for not using state-of-the-art technology. It therefore might appear much easier to publish a neural approach than other works, even though the true value of these complex models is not always fully clear. Furthermore, with the thinness of the reviewer pool, the "mathiness" of many papers [42] and the sheer amount of machine learning papers submitted every year, it becomes more and more difficult to identify those works that truly move the field forward.

One particularly seductive aspect in the context of algorithmic works on *top-n* recommendation is that there is an implicit general agreement about how research in this area should be done. Unlike in other types of recommender systems research, e.g., research that involves user studies, the experimental offline evaluation design approach is generally pre-determined and is not questioned. Generally, to have a paper accepted at one of the prestigious conferences considered in this article, one has to (at least) propose a new technical approach that outperforms some state-of-the-art methods on one evaluation protocol and on at least a couple of established metrics and publicly available datasets. Which dataset is used often seems arbitrary, and today's papers in most cases do not motivate their work based on domain-specific considerations or an underlying theory.[42] Researchers are also flexible in terms of their evaluation method. As discussed above, various protocol variants and accuracy measures are used today, and in most papers the selection of measures is not explained. Finally, as discussed throughout the article, it is difficult to know what represents the state-of-the-art in a certain area or for a specific subproblem.

All these degrees of freedom make it very tempting for researchers to focus on accuracy improvements for *top-n* recommendation, where it is in some sense easy to "prove" progress and where no special research design has to be developed and argued for. However, the established research model, as discussed, allows for a lot of arbitrariness. For example, a novel algorithm may be compared against a baseline using a number of accuracy metrics at different cut-off lengths on multiple datasets. The probability of finding some combinations of measures and datasets by which the novel algorithm seems to outperform the baseline increases with the number of cases examined [28].

Online machine learning competitions as hosted, e.g., on Kaggle,[43] represent the other extreme. On Kaggle and also in specific recommender systems challenges, the dataset and the evaluation method are pre-defined by the organization running the competition. Furthermore, participants do not see the test set and usually do not measure the recommendation quality by themselves, which avoids, as we have discovered in our study, a potential source of mistakes. One typical criticism of such online competitions is that they result in a "leaderboard chasing" culture, which can lead to limited innovation.

All in all, current research practice can easily create an illusion of progress in terms of accuracy improvements. But even when this progress is real, a further obstacle lies in us not even knowing with certainty whether these accuracy improvements will lead to better recommendations in practice, neither in terms of business value nor in terms of the quality perception by users. In fact, a number of research works indicate that algorithms with higher offline accuracy do not necessarily lead to better perceived recommendation quality [4, 17, 27, 47, 59]. For this reason, recommender systems research is different from other areas like image recognition or natural language processing (e.g., automated translation), where accuracy improvements can often directly lead to better systems.

This problem of focusing solely on accuracy optimization has actually been well known in the recommender systems community for many years [49], and the "*hyper-focus on abstract metrics*" is also common to other areas of applied machine learning research [72]. As shown in comparative evaluations like [44], however, there usually *is no best model*, and the ranking of algorithms depends on many factors like dataset characteristics or evaluation approach. While there is no doubt that being better able to predict the relevance of items for individual users is something

[42] See also Reference [35] for a satirical discussion of this general problem in machine learning.
[43] https://www.kaggle.com.

useful, considering only abstract accuracy measures appears to be a strong oversimplification of the problem.

5.4 Reproducibility and Progress: Guidelines and Best Practices

The analyses in this article point to several common issues that may lead to limited reproducibility and progress. Based on these observations, we provide a set of guidelines and best practices for algorithms research in recommender systems. The guidelines relate both to methodological aspects, i.e., how to ensure that progress is made, and publication aspects, i.e., which information should be reported in research papers and which materials should be provided publicly. Moreover, we provide recommendations for journal editors and conference chairs to improve reproducibility and progress in the discipline.

5.4.1 Methodology. Generally, researchers should select a wide, diverse and optimized set of baselines to reduce the chance of reporting progress that is only virtual.

Include baselines algorithms from different families. In our results there was no individual algorithm, or even no single family of algorithms (e.g., neighborhood-based, machine-learning, non-personalized), which was superior to the others in all experimental configurations. Therefore, researchers should strive to include well-optimized baseline algorithms of different families in their experiments and not limit themselves to a certain class of algorithms, e.g., deep learning baselines.

Systematically optimize all algorithms. All algorithms in the comparison must be optimized for each tested dataset and for the given evaluation protocol and metrics. A comparison between optimized and non-optimized algorithms is meaningless. It cannot inform us if a new method advances the state of the art. The optimization must include all relevant parameters and all baselines, even the simplest, including, for example, the *shrink term* for neighborhood-based approaches, which is known to sometimes lead to substantial performance gains. Also the number of training epochs for learning approaches is a parameter to be tuned on validation data.

Carefully choose the hyperparameter optimization strategy. Another issue we observed for several papers but have not highlighted so far is that authors often do not report the details of the applied hyperparameter optimization strategy. The chosen strategy can, however, have a significant impact on the results. The commonly used Grid Search approach has certain limitations, in particular that the results may be strongly affected by how the value ranges of continuous valued hyperparameters are discretized. Since even Random Search has shown to be often favorable over Grid search [6], the use of Random search or Bayesian strategies is recommended.

5.4.2 Reproducibility. As pointed out by Cockburn et al. in Reference [14], the pressure to publish academic works encourages researchers to consciously or unconsciously adopt not well justified methodologies—such as excluding certain users or items from the datasets—to make their findings look stronger. While such decisions can be entirely appropriate if properly motivated, engaging in selective reporting of results is not: flexible data analysis and selective reporting can dramatically increase the risk of only virtual progress. For example, if a researcher collects 10 accuracy metrics and only reports the significant ones (with significance level at 0.05), then the probability of reporting a progress that is only virtual jumps from 5% to 40%.

While in principle a paper should contain all necessary information to reproduce the results, this is often conflicting with the space limitations most publication outlets enforce. The following is a list of details that should be reported as a minimum in a paper and it is based on *The Machine Learning Reproducibility Checklist*[44] adopted, e.g., by NeurIPS '18 and NeurIPS '19:

State underlying assumptions. The assumptions the model relies upon (e.g., data properties) should be clearly stated and, when necessary, verified to be true for the evaluation setup at hand. For example, the claim that interaction maps derived from the outer product of embeddings are equivalent to images and that CNNs allow to model the embedding correlations (CoupledCF [79] and ConvNCF [29]) was not directly verified in the original articles and was later questioned by Ferrari Dacrema et al. [24].

Be clear about data pre-processing. All pre-processing steps should be described and the removal of data should be clearly motivated. In some of the articles we analyzed (DMF [77] and NeuRec [80]), pre-processing reduced the size of the dataset by more than 80%. Drastic subsampling of the dataset may entirely change its properties.

Report hyperparameter optimization details. The article should either explicitly mention or provide a reference to the range and distribution of each hyperparameter, and should report the method that was used to select them. This includes the number of epochs the model was trained for and early-stopping criteria.

Given that small details in the implementation of complex models can matter a lot, what published in a paper alone is often not sufficient to ensure reproducibility of results. Therefore, it is important for authors to also provide a publicly available implementation. The following guidelines are partially based on those adopted by NeurIPS.[45] In essence, it should be made as easy as possible for other researchers to redo the experiments.

Publish the source code of all models, including the baselines. We often observed in our analysis that researchers only published the core training and prediction routine of the newly proposed model. We could not find any article providing the source code for the baselines, when those were not already publicly available as part of other projects, and only few published the source code for both the pre-processing and splitting of the datasets (Mult-VAE [38] and NeuRec [80]).

Provide the datasets. The used datasets, or pointers to them, should be published. If possible, then the train and test splits should be published as well.

Make the reproduction easy for others. Authors should specify all software requirements and dependencies of their software, including the specific version. Provide installation scripts and machine-readable dependency descriptions. It is also desirable to provide scripts that execute the entire experimental pipeline, from pre-processing, to training, evaluation, and collection of metrics.

Use persistent repositories. Institutional repositories or DOIs should be preferred to personal webpages, since the latter can have a much shorter lifespan making previously published links unavailable [70].[46]

5.4.3 Mitigation. In the spirit of mitigating the current issues in phantom progress, our recommendation to journal editors and conference chairs is to promote submissions replicating previous

[44] https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist-v2.0.pdf.

[45] https://github.com/paperswithcode/releasing-research-code.

[46] For example, at the time of our analysis we found that the provided links to two datasets (Tafeng and Frappe) were not active anymore. We could only use these datasets, because other researchers had uploaded them independently as part of other publications.

studies. Replications of studies contribute to elevate confidence and robustness in findings beyond what is possible from a single study.

However, the main obstacle to reproducibility studies is a publication bias in which reproducibility papers are accepted for publication at a much lower rate than those providing novel results [14]. Therefore, we encourage editors and chairs to educate reviewers on the research value of replicating previous studies.

6 CONCLUSION

Our work reveals that despite the large number of new methods that are published on the topic of *top-n* recommendation every year, the progress in this field may actually seem limited, at least when considering the papers analyzed in this work. Our analysis shows that sometimes even relatively simple methods lead to performance levels that are similar or even better than the most recent complex neural methods. In other cases, the computationally expensive neural methods did not outperform well-established matrix factorization approaches or linear models. A number of issues contribute to the observed phenomena. We not only identified different methodological problems but also found that the choice of the baselines and often the lack of a proper optimization of these baselines represent key issues that hamper our progress. These phenomena however are not specific to the domain of recommender systems or to neural approaches. In the context of the use of deep learning techniques for the important *top-n* recommendation task, our analyses in some ways indicate that the power of deep learning has not been leveraged yet to the full extent as it was done in other applications areas. This, as a result, leads to a number of avenues for future research on neural methods for recommender systems.

Increasing the reproducibility of published research was identified as one of possible strategies to mitigate some of the observed problems. In the information retrieval research community, questions of replicability and reproducibility have recently received more attention, probably thanks in part to the surprising results from References [3, 39, 78]. An increased awareness and corresponding initiatives are also advisable for the recommender systems community. However, even with better reproducibility, fundamental problems of algorithms-based recommender systems research remain. The reason is that, unlike in some IR tasks, better retrieval or prediction performance does not necessarily lead to recommendations that are better in terms of the users' quality perception or in terms of the business value for providers.

In the end, these problems lead to a certain stagnation of the research field. A large number of researchers hunt for the "best" model, even though there are many indications that no such model exists, as the performance ranking of algorithms depends on many factors. Overall, a paradigmatic shift is therefore required in terms of how we evaluate recommender systems. In the future, it is, however, not only important to address the methodological issues observed in this article. We also should reconsider how much we can actually learn from offline experiments and computational accuracy measures alone. This, for example, calls for multi-method evaluation approaches that *(i)* consider the human in the loop and *(ii)* for research works that are more based on theoretical considerations than on network architecture engineering.

APPENDIX

A DETAILED ANALYSIS FOR REPRODUCIBLE ARTICLES

In this part of the Appendix, we discuss the analyzed papers in more detail and provide selected numerical results.

A.1 Collaborative Deep Learning for Recommender Systems (CDL)

Datasets. The evaluation in the original paper is based on three datasets. Two of them are data samples that were previously collected from the *CiteULike* platform and which were already used in earlier research. One dataset is based on rating data from the Netflix Prize competition, augmented with content information by the authors. In our evaluation, we considered only the CiteULike datasets, because the content information used in combination with the Netflix dataset is not publicly available. Two versions of the CiteULike dataset were considered, a dense version *CiteULike-a* and a sparser one *CiteULike-t*. Both datasets are relatively small (135k and 205k interactions, respectively). For each of these datasets, experiments were made in two sparsity configurations. These configurations are described by a parameter P, which defines how many interactions per user are left in the training set (with the rest going to the test set). For parameter P, values 1 and 10 were reported, which correspond to 5.5k and 55.5k training interactions, respectively. Note that with $P = 1$ there is only one training interaction per user in the training dataset.

Evaluation. Several baseline techniques were explored, among them a number of hybrid matrix factorization approaches, a content-based deep learning technique designed for music recommendation, as well as Collaborative Topic Regression , a method combining Latent Dirichlet Allocation on the content and collaborative filtering on the interactions. For evaluation purposes, P interactions for each user were randomly sampled to be part of the training set as mentioned above. The average results of five evaluation runs are reported. The authors report Recall for comparably long list lengths (50 to 300), and Mean Average Precision for list length 300.

Results and Discussion. The authors found their method to outperform all baselines on all measures. We could reproduce their results based on the provided code and dataset.[47] To optimize the baselines in our own evaluation, we used 20% of the training set as a validation set.[48] After optimization, our results show that CDL—in three of four configurations (CiteULike-a with *P=10* and CiteUlike-t with *P=1* and *P=10*)—was consistently outperformed by our simple hybrid technique (ItemKNN CFCBF) and, in many cases, also by the pure content-based method (ItemKNN CBF). Only when removing all but one user interaction from the CiteULike-a dataset (with *P=1*) CDL was, by a large extent, better than any of our baselines. In particular, in the settings where *P=1*, pure collaborative filtering techniques were, as expected, not competitive.

Table 5 shows exemplary results for the CiteULike-a dataset (*P=10*), with about 55k interactions in the training dataset. Detailed results for all datasets can be found in the online appendix (Section 3). In the table, we highlight in bold those entries where a baseline outperformed CDL. We can observe that, for shorter and much more typical list lengths, even the simplest collaborative filtering approaches outperform CDL. The iALS method based on matrix factorization for implicit feedback data was better with respect to CDL in all measurements and cutoff lengths. Finally, the best results were achieved with the pure content-based method that uses only item features to recommend similar items (ItenKNN CBF).

[47] In the source code the authors provided it is reported that the original evaluation contained an error such that the absolute values of the evaluation metrics was higher than the correct one, although the relative performance ordering of the algorithms remained unaltered. Once this error is fixed we can reproduce their results.

[48] Information about the validation set size was not provided in the original paper. In the evaluation scenario where *P=1*, due to the presence of only 1 training instance per user, any sampling would result in cold users. Therefore, in this scenario, the validation data is also contained in the train data. In the evaluation scenario where *P=10*, training and validation data are disjoint.

Table 5. Selected Results for the CDL Method on the CiteULike-a Dataset with $P=10$

	REC@50	REC@100	REC@150	REC@200	REC@250	REC@300
TopPopular	0.0040	0.0078	0.0103	0.0204	0.0230	0.0258
UserKNN CF jaccard	**0.0806**	**0.1207**	0.1480	0.1705	0.1887	0.2034
ItemKNN CF cosine	**0.0989**	**0.1441**	**0.1752**	**0.1982**	0.2156	0.2300
$P^3\alpha$	**0.0907**	**0.1341**	**0.1636**	0.1865	0.2055	0.2206
$RP^3\beta$[49]	**0.0963**	**0.1408**	**0.1692**	0.1908	0.2090	0.2239
$EASE^R$	**0.0839**	**0.1253**	0.1546	0.1797	0.1988	0.2128
SLIM	**0.0876**	**0.1308**	**0.1583**	0.1821	0.2005	0.2165
PureSVD	**0.0715**	0.1079	0.1313	0.1491	0.1636	0.1759
iALS	**0.0779**	**0.1388**	**0.1834**	**0.2186**	**0.2472**	**0.2706**
ItemKNN CBF cosine	**0.1989**	**0.2835**	**0.3402**	**0.3844**	**0.4193**	**0.4492**
ItemKNN CFCBF cosine	**0.1858**	**0.2816**	**0.3445**	**0.3930**	**0.4335**	**0.4642**
CDL	0.0580	0.1108	0.1546	0.1946	0.2314	0.2640

We highlight in bold those entries where a baseline outperforms CDL.

From a methodological perspective, there is no indication of why comparably long list lengths were used for evaluation in the paper and why no measurements were reported for list lengths below 50, which is commonly the case in the literature.

A.2 Collaborative Variational Autoencoder (CVAE)

Datasets and Evaluation. The CVAE method is evaluated in the same way as the CDL approach, i.e., two datasets from *CiteULIke* are used and different sparsity configurations are evaluated. Likewise, the authors of CVAE measure Recall at different (long) list lengths. As an additional baseline, the authors include the CDL [73] method described in the previous section. The hyperparameters for all baseline methods were optimized using a validation set that is, however, not described.

Results and Evaluation. We could reproduce the results for CVAE.[50] Table 6 shows the results of our experiments for the CiteULike-a dataset with $P=10$, again using the same evaluation measures and protocol as used in the original paper.

The baseline results shown in Table 6 are identical to those of Table 5—as they were done on the same dataset and with the same evaluation protocol—except that Table 6 has an additional row for the results for the CVAE method. Again, the simple hybrid baselines outperform the more complex CVAE method on all measures on this dataset. We can, however, observe that CVAE is indeed consistently better than the CDL method, which is the main baseline method in Reference [37]. For the other dataset and sparsity configurations, our results are similar to what was reported in the previous section on CDL.

Overall, the authors of CVAE could show an advance with respect to CDL, but our results indicate that CDL did not represent a strong baseline method. In the remainder of this article, we will observe the following phenomenon several times: a neural method is introduced as improving the state-of-the-art, and subsequent works only focus on outperforming this new neural method, without considering alternative baselines.

[49] We report $RP^3\beta$ [54] for completeness although the DL algorithm we evaluate here predates its publication.

[50] In the source code the authors provided it is reported that the original evaluation contained an error such that the absolute values of the evaluation metrics was higher than the correct one, although the relative performance ordering of the algorithms remained unaltered. Once this error is fixed, we can reproduce their results.

Table 6. Experimental Results for the CVAE Method for the CiteULike-a with *P=10*

	REC@50	REC@100	REC@150	REC@200	REC@250	REC@300
TopPopular	0.0040	0.0078	0.0103	0.0204	0.0230	0.0258
UserKNN CF jaccard	**0.0806**	0.1207	0.1480	0.1705	0.1887	0.2034
ItemKNN CF cosine	**0.0989**	0.1441	0.1752	0.1982	0.2156	0.2300
$P^3\alpha$	**0.0907**	0.1341	0.1636	0.1865	0.2055	0.2206
$RP^3\beta$[51]	**0.0963**	0.1408	0.1692	0.1908	0.2090	0.2239
$EASE^R$	**0.0839**	0.1253	0.1546	0.1797	0.1988	0.2128
SLIM	**0.0876**	0.1308	0.1583	0.1821	0.2005	0.2165
PureSVD	0.0715	0.1079	0.1313	0.1491	0.1636	0.1759
iALS	0.0779	0.1388	0.1834	0.2186	0.2472	0.2706
ItemKNN CBF cosine	**0.1989**	**0.2835**	**0.3402**	**0.3844**	**0.4193**	**0.4492**
ItemKNN CFCBF cosine	**0.1858**	**0.2816**	**0.3445**	**0.3930**	**0.4335**	**0.4642**
CVAE	0.0805	0.1569	0.2232	0.2760	0.3250	0.3687
CDL	0.0580	0.1108	0.1546	0.1946	0.2314	0.2640

We highlight in bold those entries where a baseline outperforms CVAE.

A.3 Neural Collaborative Filtering (NCF)

Datasets. Two datasets were used for evaluating the method, one rating dataset from MovieLens (*MovieLens1M*)[52] and one dataset with implicit feedback from *Pinterest.*[53] The Pinterest dataset was pre-processed by removing all users with less than 20 interactions. After pre-processing, the dataset contained 1.5 million interactions. For the MovieLens rating dataset, all 1 million ratings were transformed to 1 to mimic an implicit-feedback dataset, with missing entries transformed to 0.

Evaluation. The authors use a leave-last-out procedure to evaluate their method. For each user, the last interaction (based on its timestamp) is put into the test set. The resulting data splits used in the experiments are shared by the authors. To avoid to compute scores for all recommendable items, which is considered too time-consuming by the authors even for datasets of modest size, the performance of the algorithms is measured by determining how the last hidden item is ranked within other 100 randomly sampled items. Hit Rate (HR) and NDCG at list length 10 are used as performance metrics [79].

As personalized baselines, the authors include Matrix Factorization with Bayesian Personalized Ranking (BPR) matrix factorization, the eALS method from 2016 and the ItemKNN method. The original hyperparameter optimization is done on a validation set obtained by randomly selecting one interaction per user. For the ItemKNN method, the number of neighbors was varied, but no other configurations were tested by the authors (e.g., shrink term or normalization). According to the reported experiments, the NCF method, and in particular the NeuMF variant, outperforms all baselines on all dataset on all performance measures.

Results and Discussion. We could reproduce the reported results. However, the analysis of the provided source code shows that the number of training epochs was chosen by maximising the Hit Rate on the test data. Since the number of epochs is a parameter like any other, it must be fixed before testing, e.g., through early stopping on a validation set. In our experiments, for each

[51] We report $RP^3\beta$ [54] for completeness although the DL algorithm we evaluate here predates its publication.

[52] https://grouplens.org/datasets/movielens/.

[53] https://sites.google.com/site/xueatalphabeta/academic-projects.

Table 7. Experimental Results for NCF (MovieLens 1M)

	HR@1	NDCG@1	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1051	0.1051	0.3048	0.2064	0.4533	0.2542
UserKNN CF asymmetric	0.1921	0.1921	0.5070	0.3546	0.6768	0.4100
ItemKNN CF asymmetric	0.1843	0.1843	0.4906	0.3400	0.6627	0.3956
$P^3\alpha$	0.1791	0.1791	0.4846	0.3352	0.6460	0.3876
$RP^3\beta$[54]	0.1836	0.1836	0.4935	0.3419	0.6758	0.4011
$EASE^R$	**0.2225**	**0.2225**	**0.5629**	**0.3986**	**0.7192**	**0.4494**
SLIM	**0.2207**	**0.2207**	**0.5576**	**0.3953**	**0.7162**	**0.4468**
PureSVD	**0.2132**	**0.2132**	0.5339	0.3783	0.6937	0.4303
iALS	**0.2106**	**0.2106**	**0.5505**	**0.3862**	**0.7109**	**0.4382**
NCF (NeuMF variant)	0.2088	0.2088	0.5411	0.3803	0.7093	0.4349

We highlight in bold those entries where a baseline outperforms NCF.

Table 8. Experimental Results for NCF (Pinterest)

	HR@1	NDCG@1	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.0467	0.0467	0.1665	0.1064	0.2740	0.1409
UserKNN CF jaccard	**0.2898**	**0.2898**	0.7038	**0.5056**	0.8655	**0.5583**
ItemKNN CF asymmetric	**0.2903**	**0.2903**	**0.7117**	**0.5096**	0.8766	**0.5633**
$P^3\alpha$	**0.2853**	**0.2853**	0.7022	0.5024	0.8700	0.5571
$RP^3\beta$[54]	**0.2966**	**0.2966**	**0.7151**	**0.5149**	**0.8796**	**0.5685**
$EASE^R$	**0.2909**	**0.2909**	0.7070	**0.5077**	0.8684	**0.5604**
SLIM	**0.2913**	**0.2913**	0.7059	**0.5072**	0.8679	**0.5601**
PureSVD	0.2630	0.2630	0.6628	0.4706	0.8268	0.5241
iALS	**0.2811**	**0.2811**	**0.7144**	**0.5061**	0.8761	**0.5590**
NCF (NeuMF variant)	0.2801	0.2801	0.7101	0.5029	0.8777	0.5576

We highlight in bold those entries where a baseline outperforms NCF.

algorithm (including NCF), we therefore report the performance measure for the number of epochs that was considered optimal based on the validation set.

Tables 7 and 8 report our results for both the MovieLens and Pinterest datasets. We report the results for list length 10, as in the original paper. Since the authors in Reference [30] also plot the results at different list lengths from 1 to 10, we also include measurements at list lengths 1 and 5 for comparison purposes.

On the well-known MovieLens dataset (Table 7), NeuMF was competitive against the simple baselines, however was outperformed by all but one non-neural machine learning methods. On the Pinterest dataset (Table 8), NeuMF could only outperform PureSVD, which is not optimized for implicit feedback datasets. Most non-neural machine learning techniques, were often either similar or better than NeuMF.[55] These findings have been confirmed in a recent article by Rendle

[54] We report $RP^3\beta$ [54] for completeness although the DL algorithm we evaluate here predates its publication.

[55] It shall be noted here that after the first publication of our results [23], the authors of NeuMF provided us with an alternative configuration of their method, which included new hyperparameter values taken from alternative hyperparameter ranges, and requiring other slight changes in the training procedure. While this new configuration led to slightly improved results for their method, the results of our analysis were confirmed. In this context, we would like to clarify that for all neural methods investigated here better configurations than those reported in the original papers may indeed exist. Finding such configurations, e.g., in the form of better hyperparameter ranges or alternative network structures, is, however, not

et al. [56], which discussed the problem of learning a dot product with multilayer perceptrons. The article shows that this is not a trivial task and that the computation cost of using a neural model can be unpractical for production environments where efficient algorithms based on the dot product are available.

As a side observation, we can see that machine learning methods were clearly favorable over simple techniques for the MovieLens dataset. For the Pinterest dataset, however, it turns out that this advantage diminishes—at least in this experiment—and that a well-tuned ItemKNN method led to similar and sometimes better performance than machine learning techniques.

A.4 Deep Matrix Factorization (DMF)

Datasets. Experiments were made on four public datasets: the two smallest MovieLens datasets (*100k* and *1M*),[56] and two publicly available datasets collected from Amazon.com (for the *Movie* and *Music* domains).[57] All datasets contain ratings on a 1 to 5 scale. The datasets were pre-processed (if needed) so that there were at least 20 ratings for each user. Furthermore, for the Amazon datasets only, items were considered for which more than 5 ratings existed.

The Amazon Music data set that resulted from the pre-processing step was shared by the authors. It however contains users with less than 20 interactions and items with less than 5 ratings. Therefore, it remains unclear how exactly the filtering was done. To keep the results presented in this article consistent across datasets, we have pre-processed all datasets—including Amazon Music—as described in the original paper, and we have not used the Amazon Music dataset shared by the authors. We also run our experiments on the Amazon Music dataset shared by the authors: the relative performance between baselines and DMF (not reported here) do not change with respect to the results reported here.

Evaluation. The evaluation procedure is *leave-last-out* similar to the one used for the NCF method. For each user, the last interaction (based on its timestamp) is held out and ranked together with 99 negative (non-interacted) random items.[58] The Hit Rate and NDCG at list length 10 are used as metrics. The data splits that were used in the experiments were not shared by the authors; therefore, we created data splits based on the information in the paper.

As personalized baselines, the authors consider NCF [30] as well as the baselines reported in that article, i.e., eALS and ItemKNN. However, the authors used NCF with binarized feedback, while DMF used explicit feedback. Hyperparameters for the machine learning methods were tuned on a validation set built from the training set by randomly sampling one interaction per user, and the authors report that eALS and NCF were tuned as in the original papers. For the ItemKNN method, no details about neighborhood sizes or the used similarity function are provided.

Results and Discussion. We reproduced the experiments reported by the authors based on the code that was provided to us upon request. We ran DMF with both loss functions that were also evaluated in the paper but could not confirm that the normalized version *nce* leads to accuracy improvements over the binary version *bc*. Our results however revealed that for three of four datasets one of the simple baselines outperformed DMF on both measures.

Table 9 shows the results for the MovieLens datasets, which are among the most often used ones in the literature. The results obtained by DMF are better than traditional nearest-neighbor

the goal of our work. Instead, our goals are to assess the reproducibility of existing works and to compare the best reported results against existing baseline techniques.

[56] https://grouplens.org/datasets/movielens/.

[57] http://jmcauley.ucsd.edu/data/amazon/.

[58] The paper reports that 100 negative items are used, as described for NCF. However, the source code provided by the authors uses 99 negative items. In our experiments, we have used 99 negative items.

Table 9. Experimental Results for DMF for the MovieLens1M (Left) and MovieLens100k (Right) Datasets

	HR@10	NDCG@10
TopPopular	0.4418	0.2475
UserKNN CF asymmetric	0.6324	0.3779
ItemKNN CF cosine	0.6347	0.3808
P$^3\alpha$	0.6097	0.3639
RP$^3\beta$[59]	0.6304	0.3726
EASE^R	0.6693	**0.4100**
SLIM	**0.6825**	**0.4209**
PureSVD	0.6570	0.4015
iALS	**0.6947**	**0.4257**
DMF *nce*	0.6266	0.3768
DMF *bc*	0.6731	0.4033

	HR@10	NDCG@10
TopPopular	0.4145	0.2342
UserKNN CF asymmetric	0.5994	0.3492
ItemKNN CF tversky	0.6026	0.3506
P$^3\alpha$	0.5717	0.3421
RP$^3\beta$[59]	0.5685	0.3270
EASE ^{R}	**0.6089**	0.3571
SLIM	**0.6238**	**0.3765**
PureSVD	0.5877	0.3555
iALS	**0.6142**	**0.3691**
DMF *nce*	0.5930	0.3410
DMF *bc*	0.6026	0.3623

We highlight in bold those entries where a baseline outperforms DMF.

Table 10. Experimental Results for DMF for the Amazon Music (Left) and Amazon Movies (Right) Datasets

	HR@10	NDCG@10
TopPopular	0.5308	0.3037
UserKNN CF cosine	**0.6694**	0.4798
ItemKNN CF cosine	0.6647	**0.4880**
P$^3\alpha$	0.6588	**0.4823**
RP$^3\beta$[59]	**0.6754**	**0.4912**
EASE^R	0.6600	**0.4836**
SLIM	0.6469	0.4744
PureSVD	0.5912	0.4190
iALS	0.6600	**0.4880**
DMF nce	0.4799	0.3371
DMF bc	0.6659	0.4815

	HR@10	NDCG@10
TopPopular	0.5794	0.3489
UserKNN CF cosine	0.7327	0.5132
ItemKNN CF asymmetric	0.6986	0.4914
P$^3\alpha$	0.6972	0.5028
RP$^3\beta$[59]	0.7107	0.5078
EASE^R	-	-
SLIM	0.6981	0.5005
PureSVD	0.6021	0.4156
iALS	0.7352	0.5230
DMF *nce*	0.6832	0.4677
DMF *bc*	**0.7818**	**0.5417**

EASE^R results for Amazon Movies are missing, because the code required too much memory. We highlight in bold those entries where a baseline outperforms DMF.

baselines on the MovieLens1M dataset, but slightly worse than those obtained with the iALS and SLIM. For the smaller MovieLens100k dataset, the observed ranking is generally similar and again iALS and SLIM outperform DMF on both measures.

The detailed results for the Amazon datasets are shown in Table 10. The results for EASE^R are missing for the Amazon Movies dataset, as the author-provided Python implementation of the method needed too much memory. For the Amazon Music dataset it is interesting to observe that the simple UserKNN and RP$^3\beta$ methods work better here than other machine learning models, and also better than DMF. For the Amazon Movies dataset, the DMF method was actually much better than all other methods on both measures. In particular the gains in terms of the Hit Rate are substantial and much higher than the second best method iALS.

Like for the case of CDL and the CVAE methods described in Sections A.1 and A.2, a better performance could only be observed for one of the datasets. In the case of CDL and CVAE better

[59] We report RP$^3\beta$ [54] for completeness although the DL algorithm we evaluate here predates its publication.

results were obtained for very sparse datasets with only one training interaction per user. Looking at the Amazon Movies dataset characteristics in the context of the DMF method, we can see that it is extremely sparse after the pre-processing step, in which 80% of the interactions were removed. In the end, there are 878k remaining interactions for over 80k movies. The Amazon Music dataset is even sparser. After the pre-processing step, which removes more than 94% of the interactions, it has only 46k interactions for 18k items. Further investigations are necessary to better understand why DMF works so well in this case, which could help us design algorithms that also work well on other datasets with similar characteristics.

Regarding methodological aspects, we found that the authors reported the best Hit Rate and NDCG results across different epochs. We therefore report the numbers here that were obtained after determining a suitable number of epochs on the validation set. In that context, the provided code shows that the authors sample different negative items to be used for testing in each training epoch. This seems questionable as well, in particular when considered in combination of the practice of reporting the best value for each metric across epochs. In our experiments, we use the same negative item set for all evaluations.

From a conceptual perspective, the authors argue that they combine implicit feedback and explicit feedback in their approach. While this might be true in some interpretation, the authors mainly rely on the explicit ratings and add zeros to the empty matrix cells. Furthermore, when comparing their method with NCF, they only fed the binarized data to NCF, even though this method could deal with explicit rating data as well.

A.5 Variational Autoencoders for Collaborative Filtering (Mult-VAE)

Datasets. The authors use three datasets for evaluation. The first two datasets contain explicit feedback in the form of movie ratings (*MovieLens20M*[60] and *Netflix*[61]). The third dataset contains play counts for musical tracks. All datasets are binarized. For the movie datasets, ratings higher than three are considered positive signals and only users with more than five interactions are retained. For the music dataset, users with more than 20 interactions are retained; tracks that were listed less than 200 times are filtered out. After pre-processing, the datasets are still relatively large, having between 10 and almost 57 millions interactions.[62]

Evaluation. Four machine learning models are used as baselines, iALS, SLIM, NCF [30] (see Section A.3) and the Collaborative Denoising Autoencoder (CDAE) method proposed in Reference [75] in 2016. For evaluation purposes, the datasets are split into training, validation and test splits by holding out users. For instance, for the MovieLens20M dataset (136k users overall), 10k users are removed for validation and 10k users are removed for testing. For each hold-out user, 80% of the interactions are used as user profile, and the remaining 20% are used as ground truth to measure the performance metrics. The models are optimized for NDCG@100 on the validation set. Performance results for Recall@20 and Recall@50 are reported as well.

Note that to be able to use matrix factorization baselines on cold users we built the cold users' latent factors based upon both their user profile and the latent factors of the warm items. In particular, we added a hyperparameter to the matrix factorization models to select how those cold user's latent factors are estimated, either via an item-based similarity or an item embeddings average; see Reference [18]. The first case is inspired by the *folding-in* technique [62] for the PureSVD model. It can be mathematically derived that PureSVD has an equivalent ItemKNN formulation,

[60] https://grouplens.org/datasets/movielens/.

[61] https://www.kaggle.com/netflix-inc/netflix-prize-data.

[62] We did not run experiments for the music dataset as the original paper did not contain sufficient information to guarantee we used the dataset in the exact same way as the authors.

Table 11. Results for Mult-VAE for the MovieLens20M Dataset

	REC@20	NDCG@20	REC@50	NDCG@50	REC@100	NDCG@100
TopPopular	0.1441	0.1201	0.2320	0.1569	0.3296	0.1901
UserKNN	-	-	-	-	-	-
ItemKNN CF asymmetric	0.2937	0.2444	0.4486	0.3087	0.5709	0.3527
P$^3\alpha$	0.2620	0.2168	0.4047	0.2742	0.5287	0.3182
RP$^3\beta$	0.3006	0.2501	0.4540	0.3133	0.5797	0.3583
EASE^R	0.3530	**0.3074**	0.5147	**0.3755**	0.6353	**0.4196**
SLIM	0.3356	0.2920	0.4893	0.3576	0.6110	0.4017
PureSVD	0.2935	0.2514	0.4371	0.3117	0.5544	0.3538
iALS	0.2968	0.2496	0.4406	0.3090	0.5631	0.3521
Mult-VAE	**0.3541**	0.2988	**0.5222**	0.3690	**0.6517**	0.4158

UserKNN could not be applied because of the evaluation protocol (hold-out of users). We highlight in bold those entries where a baseline outperforms Mult-VAE.

Table 12. Results for Mult-VAE for the Netflix Dataset

	REC@20	NDCG@20	REC@50	NDCG@50	REC@100	NDCG@100
TopPopular	0.0786	0.0762	0.1643	0.1159	0.2717	0.1570
UserKNN	-	-	-	-	-	-
ItemKNN CF cosine	0.2091	0.1970	0.3387	0.2592	0.4598	0.3092
P$^3\alpha$	0.1960	0.1759	0.3325	0.2412	0.4633	0.2962
RP$^3\beta$	0.2210	0.2053	0.3633	0.2739	0.4932	0.3281
EASE^R	**0.2681**	**0.2591**	**0.4170**	**0.3334**	**0.5471**	**0.3890**
SLIM	0.2555	**0.2479**	0.4002	**0.3203**	0.5299	**0.3752**
PureSVD	0.2271	0.2184	0.3593	0.2840	0.4784	0.3342
iALS	0.1956	0.1839	0.3138	0.2410	0.4216	0.2862
Mult-VAE	0.2615	0.2423	0.4127	0.3167	0.5456	0.3730

UserKNN could not be applied because of the evaluation protocol (hold-out of users). We highlight in bold those entries where a baseline outperforms Mult-VAE.

in which the similarity matrix is the product of the items' latent factor matrix by its transpose. In this case a further parameter is introduced to control the number of neighbors, as in the other neighborhood-based methods. In the second case the latent factors of a user are the product of the user profile and the items' latent factors, resulting in the average of the embeddings of the items the user interacted with. Both methods proved effective.

Results and Discussion. Using the code, the data splits and information about the seed for the random number generator that the authors provided, we could reproduce the results from the paper. In the original paper NDCG@100 is used as an optimization goal, however no reason is provided for this, as well as why Recall@20 and Recall@50 are used as additional measures, but not, e.g., Recall@100. To obtain a more comprehensive picture, we made additional measurements at the corresponding but missing cut-off values: Recall@100, NDCG@20 and NDCG@50. Table 11 shows our results for the MovieLens20M dataset and Table 12 those for the Netflix data.

For the MovieLens dataset, we observed a positive result and could confirm the claims made by the authors of Mult-VAE. On all measurements, both the original and the additional ones, Mult-VAE leads to better performance results than all baseline methods that were available at the time the algorithm was proposed. SLIM is the second best method in this evaluation, with performance results that are around 1% to 2% lower in terms of the NDCG.

For the Netflix dataset, the claims of the authors could not be confirmed to the full extent. In terms of NDCG, which is the optimization criterion, SLIM outperforms Mult-VAE at all list lengths. Mult-VAE is however better in terms of Recall.

Overall, with Mult-VAE a method was found that was easy to reproduce, thanks to all needed material being made by the authors publicly available. Furthermore, as our results indicated, the method consistently outperformed previous methods at least on one well-known and comparably large dataset. We could also confirm that EASE^R leads to improvements over Mult-VAE in most cases, supporting the claim that shallow models are a competitive solution.

A.6 NeuRec: On Nonlinear Transformation for Personalized Ranking

Datasets. The authors use four public datasets for their evaluations (*MovieLens1M,*[63] *HetRec,*[64] *FilmTrust,*[65] *Frappe*[66]). Three of them contain movie ratings, which are binarized by converting all ratings to 1 and the missing entries to 0. The largest dataset (MovieLens1M) comprises 1 million interactions. The fourth dataset (Frappe) is from the domain of mobile app recommendation and contains about 100k interactions, reduced to 20k after pre-processing, which consisted of the removal of multiple interactions between the same user-item pairs. Data splits were not provided online but could be reproduced based on the information in the paper.

Evaluation. The authors use five random training-test splits (80%/20%) for evaluation and report the average results. As performance metrics, the authors use Precision and Recall at list lengths 5 and 10, as well as Mean Average Precision, MRR, and the NDCG, at list length 10.

As non-trivial baselines, the authors consider SLIM, BPR matrix factorization, NeuMF and the GMF model, which is part of NCF [30]. Information about hyperparameter tuning for the baselines is not provided except for GMF and NeuMF, which are said to use the "default" configuration as described in the original article. Hyperparameters for NeuRec were determined through grid search and the finally used values are reported in the paper in detail. The number of training epochs is not reported in the paper.[67] As usual, we selected the number of epochs via early stopping on a validation split.

Results and Discussion. Even though the authors published a runnable implementation of their method and provided detailed information on the hyperparameters, we could *not* obtain the results reported in the original paper. We contacted the authors, but we were not able to reconstruct an experimental pipeline (from pre-processing to hyperparameter optimization) that led to results that were comparable to the ones reported in the original paper. In the end, the reason for this discrepancy could not be clarified. The outcome of our evaluation is that NeuRec is outperformed on any data set and almost on any measure by at least one, but usually several, of the baselines in our comparison.

Since the detailed results are comprehensive, given the number of datasets and evaluation measures, we only provide in Table 13 the results for the most commonly used MovieLens1M dataset and for list lengths 5 and 10. All other results can again be found in the online appendix (Section 3).

[63] https://grouplens.org/datasets/movielens/.

[64] https://grouplens.org/datasets/hetrec-2011/.

[65] https://www.librec.net/datasets/filmtrust.zip.

[66] The original download link was not active anymore at the time or our analysis. We provide an alternative download link https://github.com/hexiangnan/neural_factorization_machine/archive/master.zip.

[67] According to an exchange of emails with the authors, the training was done for a large number of epochs and the best performance values on the test set were reported.

Table 13. Experimental Results for NeuRec for the MovieLens1M Dataset

	@5					@10				
	PREC	REC	MAP	NDCG	MRR	PREC	REC	MAP	NDCG	MRR
TopPopular	0.2105	0.0402	0.1531	0.0689	0.3621	0.1832	0.0685	0.1168	0.0939	0.3793
UserKNN CF asymmetric	**0.4212**	**0.1065**	**0.3441**	**0.1674**	**0.6399**	**0.3617**	**0.1726**	**0.2774**	**0.2230**	**0.6509**
ItemKNN CF asymmetric	**0.3995**	**0.0984**	**0.3244**	**0.1563**	**0.6179**	**0.3452**	**0.1590**	**0.2618**	**0.2084**	**0.6293**
$P^3\alpha$	**0.4041**	**0.1007**	**0.3286**	**0.1596**	**0.6250**	**0.3456**	**0.1627**	**0.2627**	**0.2121**	**0.6362**
$RP^3\beta$	**0.4080**	**0.1007**	**0.3325**	**0.1602**	**0.6260**	**0.3508**	**0.1639**	**0.2676**	**0.2137**	**0.6374**
$EASE^R$	**0.4488**	**0.1134**	**0.3717**	**0.1779**	**0.6620**	**0.3857**	**0.1820**	**0.3035**	**0.2364**	**0.6717**
SLIM	**0.4437**	**0.1106**	**0.3692**	**0.1749**	**0.6578**	**0.3813**	**0.1770**	**0.3003**	**0.2321**	**0.6679**
PureSVD	**0.4123**	**0.0987**	**0.3371**	**0.1586**	**0.6266**	**0.3575**	**0.1624**	**0.2722**	**0.2132**	**0.6380**
iALS	**0.4164**	**0.1036**	**0.3373**	**0.1635**	**0.6327**	**0.3628**	**0.1702**	**0.2743**	**0.2200**	**0.6443**
INeuRec	0.3280	0.0663	0.2554	0.1110	0.5003	0.2839	0.1094	0.2027	0.1500	0.5129
UNeuRec	0.2098	0.0395	0.1560	0.0684	0.3663	0.1856	0.0688	0.1199	0.0944	0.3852

We highlight in bold those entries where a baseline outperforms NeuRec.

Looking at the results, we observe that on the MovieLens dataset even the simplest baselines are better than NeuRec and that the performance of the best baselines is better by a large margin. For the HetRec and FilmTrust datasets (not shown in detail here), the picture is mostly the same. Finally, for the small and rarely used Frappe dataset, NeuRec actually leads to the best results for Precision@5, but is outperformed, e.g., by $RP^3\beta$ on all other measures.[68]

Regarding methodological aspects, we found again that researchers optimized the number of epochs on the test set and apparently reported the best results of NeuRec for different measures at, potentially, different training epochs. Furthermore, it is unclear from the paper if hyperparameter optimization was done for the baselines. For the NCF method, the authors state that they used the "configuration" proposed in the original paper, but it is unclear if this refers to the network structure, the hyperparameters, or both.

A.7 CoupledCF: Learning Explicit and Implicit User-item Couplings

Datasets. Experiments were made on two public datasets, the *MovieLens1M*[69] rating dataset and a dataset called *Tafeng*[70] containing grocery store transactions. The Tafeng dataset has about 750k transactions (i.e., less than the MovieLens1M dataset), but is much more sparse as it contains many more users and items. The explicit ratings in the MovieLens1M dataset are transformed into binary ratings, where each rating is considered as a positive interaction. Both datasets contain side information about users and items that is used by the CoupledCF algorithm. Therefore, we have included item-based and user-based content techniques among the baselines. The authors of the paper provided us with the train-test splits, including the sampled test negative items they had used during the evaluation.

Evaluation. A leave-one-out procedure is used, where for each user one random interaction is put in the test set. For evaluating the performance, 99 items are sampled for which there was no interaction for the given user. The 100 items are then ranked by the algorithm and the Hit Rate

[68] We point out that the Frappe dataset is very small and exhibits very unstable results depending on the random split. In particular, compared to the results reported in the original paper, our TopPopular algorithm exhibits results that are four times higher; also NeuRec's values are two times higher.

[69] https://grouplens.org/datasets/movielens/.

[70] The original download link was not active anymore at the time or our analysis. We provide an alternative download link https://www.kaggle.com/chiranjivdas09/ta-feng-grocery-dataset.

and the NDCG are used to evaluate the performance. Cut-off list lengths between 1 and 10 were considered.

The hyperparameters of the proposed model were systematically fine-tuned by the authors. Information about hyperparameter tuning for the baselines is not provided. The considered baselines include NCF [30], and Google's Wide&Deep method. From the original source code, we observed that the authors reported the best results for each metric on the test data across different epochs. Since this is inappropriate, we report the values of the metrics after the optimal number of epochs is chosen with early stopping on the validation set.

Results and Discussion. We could not fully reproduce the results by the authors based on the provided code and data splits. Different variations of the proposed model were tested in the original paper. We used the best-performing one (CoupledCF) in our experiments as well as their simplest variant (DeepCF), where the former is the only one for which we could fully reproduce the results. Since there were some apparent issues regarding the way the authors did the sampling, which we discuss below, we recreated the train-test splits according to the descriptions in the paper. We also used a validation set for hyperparameter tuning. While the numerical results obtained using the split provided by the original authors and the one generated by us are different, the relative ordering of the algorithms in terms of their recommendation quality remains the same. We report the results obtained on the data split generated by us. As previously mentioned, we observe there were some apparent issues regarding the way the authors sampled the negative items. For both Movielens and Tafeng the negative item data contains duplicates leading to many users (72% for Movielens, 28% for Tafeng) having less of the desired 99 negative items, e.g., some have only 93. For the Tafeng dataset, cumulatively, almost 3,000 negative items (0.1% of the total number of negative items) also appeared as train items or test items for that same user. If the number of unique negative items is not constant, then different users will be evaluated under *slightly* different conditions, in particular, the recommendation problem becomes easier as the number of negatives decreases due to the reduction in the pool of alternatives the recommender has to choose from. We nonetheless assume that the impact of these inconsistencies will generally be small, since both the original split and the one generated by us exhibit the same ranking of the algorithms in terms of their performance. More strikingly, however, for 8% of the users in the Tafeng dataset, we observed problems regarding the usefulness of the test data. Some of these users have a test item but no negative items, meaning that there is only a single item that can be ranked and even a random recommender will exhibit perfect recommendations. Other users have negative items but no test item, meaning that no correct recommendations are possible.

Our experimental results are shown in Table 14 (MovieLens) and Table 15 (Tafeng).

For the MovieLens datasets, we can observe that CoupledCF is almost consistently able to outperform the simple neighborhood-based methods and the hybrids (except for very short list lengths). However, relatively simple non-neural methods like iALS and EASE^R are consistently better than CoupledCF. The differences between the CoupledCF and the DeepCF variant are tiny and the simpler DeepCF is sometimes even better. This stands in contrast to the results in the original article, where the differences were large.

For the Tafeng dataset, even the nearest-neighbor methods outperform CoupledCF by far. Only the pure content-based baselines do not reach the performance level of CoupledCF. Generally, on this dataset, the performance of the proposed method is at the level of the TopPopular baseline. The simpler DeepCF method also leads to better accuracy results than the CoupledCF variant.

Looking at methodological aspects, it seems that the baselines were not properly optimized and default hyperparameters were used. Furthermore, from the provided code it seems that the number of epochs was determined on the test set, as was done in other papers examined in this

Table 14. Experimental Results for CoupledCF for the MovieLens1M Dataset

	HR@1	NDCG@1	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1593	0.1593	0.4217	0.2936	0.5813	0.3451
UserKNN CF asymmetric	0.3546	0.3546	0.6914	0.5343	0.8114	0.5735
ItemKNN CF cosine	0.3305	0.3305	0.6682	0.5080	0.7940	0.5488
P$^3\alpha$	0.3316	0.3316	0.6543	0.5031	0.7687	0.5402
RP$^3\beta$	0.3464	0.3464	0.6743	0.5198	0.7959	0.5591
EASE^R	**0.4003**	**0.4003**	**0.7258**	**0.5738**	**0.8343**	**0.6093**
SLIM	**0.3906**	**0.3906**	**0.7116**	**0.5625**	**0.8315**	**0.6014**
PureSVD	**0.3735**	**0.3735**	**0.7088**	**0.5522**	0.8132	**0.5861**
iALS	**0.3816**	**0.3816**	**0.7121**	**0.5581**	0.8200	**0.5933**
ItemKNN CBF asymmetric	0.0884	0.0884	0.2586	0.1752	0.3780	0.2137
UserKNN CBF tversky	0.1714	0.1714	0.4427	0.3108	0.6065	0.3636
ItemKNN CFCBF cosine	0.3328	0.3328	0.6694	0.5107	0.7985	0.5526
UserKNN CFCBF dice	**0.3555**	**0.3555**	0.6869	0.5328	0.8008	0.5698
DeepCF	0.3550	0.3550	0.7017	0.5388	0.8272	0.5794
CoupledCF	0.3522	0.3522	0.7018	0.5374	0.8247	0.5775

We highlight in bold those entries where a baseline outperforms CoupledCF.

Table 15. Experimental Results for CoupledCF for the Tafeng Dataset

	HR@1	NDCG@1	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	**0.2654**	**0.2654**	0.5194	0.3965	0.6549	0.4402
UserKNN CF cosine	**0.3215**	**0.3215**	**0.5412**	**0.4369**	0.6415	**0.4693**
ItemKNN CF cosine	**0.3314**	**0.3314**	**0.5424**	**0.4427**	0.6376	**0.4735**
P$^3\alpha$	**0.3245**	**0.3245**	**0.5503**	**0.4437**	0.6404	**0.4730**
RP$^3\beta$	**0.3202**	**0.3202**	**0.5525**	**0.4424**	0.6470	**0.4732**
EASE^R	**0.3272**	**0.3272**	**0.5452**	**0.4417**	0.6435	**0.4736**
SLIM	**0.3233**	**0.3233**	**0.5438**	**0.4389**	0.6476	**0.4726**
PureSVD	0.2462	0.2462	0.4889	0.3714	0.6260	0.4156
ItemKNN CBF asymmetric	0.0589	0.0589	0.0958	0.0769	0.1467	0.0931
UserKNN CBF asymmetric	0.2464	0.2464	0.4654	0.3600	0.5798	0.3970
ItemKNN CFCBF asymmetric	**0.3331**	**0.3331**	**0.5434**	**0.4442**	0.6314	**0.4727**
UserKNN CFCBF asymmetric	**0.3424**	**0.3424**	**0.5882**	**0.4713**	**0.6937**	**0.5055**
DeepCF	0.2647	0.2647	0.5244	0.3995	0.6583	0.4428
CoupledCF	0.2641	0.2641	0.5175	0.3948	0.6499	0.4377

We highlight in bold those entries where a baseline outperforms CoupledCF.

work. A specific problem in this work also lies in the creation of the train and test splits, which are inconsistent with the description reported in the article and are likely the result of an erroneous splitting procedure.

Further methodological problems were discussed in a recent article by Ferrari Dacrema et al. [24], which questioned the claim that embedding maps derived from the outer product of the embeddings are analogous to images. The article shows that embedding maps do not have a semantically relevant topology and do not share image properties, therefore the use of CNNs is not well justified. Moreover, contrary to what stated in the original paper, CoupledCF does not benefit

Table 16. Experimental Results for DELF on the MovieLens Dataset

	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
TopPopular	0.3302	0.2229	0.4696	0.2674	0.6577	0.3148
UserKNN CF asymmetric	**0.5205**	**0.3635**	**0.6852**	**0.4168**	0.8329	**0.4542**
ItemKNN CF cosine	0.4936	0.3426	0.6677	0.3989	0.8243	0.4387
$P^3\alpha$	0.4945	0.3438	0.6574	0.3965	0.7952	0.4313
$RP^3\beta$	0.5138	0.3559	0.6809	0.4102	0.8276	0.4475
$EASE^R$	**0.5716**	**0.4064**	**0.7258**	**0.4566**	**0.8516**	**0.4887**
SLIM	**0.5706**	**0.4038**	**0.7306**	**0.4557**	**0.8586**	**0.4882**
PureSVD	**0.5513**	**0.3891**	**0.7021**	**0.4382**	0.8303	**0.4708**
iALS	**0.5643**	**0.3975**	**0.7228**	**0.4489**	**0.8354**	**0.4776**
DELF MLP	0.5168	0.3587	0.6809	0.4119	0.8342	0.4508
DELF EF	0.4805	0.3305	0.6504	0.3852	0.8043	0.4243

We highlight in bold those entries where a baseline outperforms DELF.

from the additional parameter space provided by the embedding correlations that can be removed from the model without altering its accuracy.

A.8 DELF: A Dual-embedding-based Deep Latent Factor Model for Recommendation

Datasets. Two public rating datasets are used for the evaluation. One is the well-known *MovieLens1M*[71] dataset and the other one the *Amazon Music*[72] dataset. The rating datasets were binarized by transforming each non-zero rating to 1. Pre-processing was applied so that for both datasets only users were retained for which more than 20 interactions were observed. Through this preprocessing, the Amazon Music dataset was reduced to less than one tenth of its original size in terms of interactions (only 76k interactions for 40k items remain in the Amazon Music dataset). The Amazon Music dataset contains, on average, only 2 interactions per item. Due to this, when the train-test split is performed according to the timestamp, 52% of the items in the test data are cold (i.e., they never appear in the train data for any user). An evaluation so strongly oriented towards cold items is particularly surprising, considering that DELF is a pure collaborative algorithm that, as such, cannot be able to learn a representation for any of those cold items.

Evaluation. The evaluation procedure was similar to the one used for NCF as discussed in Section A.3: a leave-last-out procedure was applied using the interaction timestamp, the hidden element was ranked within 99 randomly sampled negative items, and the Hit Rate and the NDCG at a cut-off length of 10 were used as performance measures. The non-trival baselines include BPR matrix-factorization, iALS, DMF as discussed in Section A.4, and two variants of the NCF model [30] described in Section A.3.

The hyperparameters for the proposed model were systematically optimized on a validation set built with the second most recent interaction. No information is provided regarding the hyperparameter optimization of the baselines.

Results and Discussion. We reproduced the results using a validation set that we constructed in the same way as reported by the authors. NDCG@10 was used as an optimization criterion.

The obtained results are shown in Table 16 (MovieLens1M) and Table 17 (Amazon Music). For the MovieLens1M dataset, all our machine learning baselines outperformed *DELF* on all measures.

[71] https://grouplens.org/datasets/movielens/.

[72] http://jmcauley.ucsd.edu/data/amazon/.

Table 17. Experimental Results for DELF for the Amazon Music Data When the Recommendation of Cold Items Is Allowed

	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
TopPopular	0.2452	0.1726	0.3057	0.1921	0.3744	0.2094
UserKNN CF cosine	0.3248	0.2544	0.3760	0.2708	0.4376	0.2864
ItemKNN CF asymmetric	0.3204	0.2566	0.3711	0.2731	0.4327	0.2886
$P^3\alpha$	0.3188	0.2524	0.3684	0.2684	0.4300	0.2839
$RP^3\beta$	0.3155	0.2494	0.3684	0.2663	0.4272	0.2811
$EASE^R$	-	-	-	-	-	-
SLIM	0.3199	0.2577	0.3678	0.2730	0.4354	0.2900
PureSVD	0.2627	0.2141	0.3084	0.2290	0.3537	0.2405
iALS	0.3319	0.2604	0.3717	0.2732	0.4229	0.2860
DELF MLP	0.2986	0.2339	0.3619	0.2542	**0.4561**	0.2778
DELF EF	**0.5422**	**0.3632**	**0.7439**	**0.4290**	**0.8578**	**0.4583**

$EASE^R$ results are missing, because the code required too much memory on these datasets. We highlight in bold those entries where a baseline outperforms DELF.

Table 18. Experimental Results for DELF for the Amazon Music Data When the Recommendation of Cold Items Is *Not* Allowed

	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
TopPopular	0.2474	0.1730	0.3041	0.1913	0.3738	0.2090
UserKNN CF cosine	**0.3150**	**0.2495**	**0.3471**	**0.2600**	0.3738	**0.2668**
ItemKNN CF asymmetric	**0.3090**	**0.2506**	**0.3401**	**0.2609**	0.3717	**0.2689**
$P^3\alpha$	**0.3074**	**0.2465**	**0.3373**	**0.2564**	0.3689	**0.2644**
$RP^3\beta$	**0.3046**	**0.2434**	**0.3379**	**0.2543**	0.3651	**0.2611**
$EASE^R$	-	-	-	-	-	-
SLIM	**0.3101**	**0.2526**	**0.3411**	**0.2625**	0.3711	**0.2701**
PureSVD	0.2627	0.2141	0.3084	0.2290	0.3542	0.2406
iALS	**0.3319**	**0.2604**	**0.3706**	**0.2729**	**0.4109**	**0.2831**
DELF MLP	0.2905	0.2239	0.3275	0.2361	0.3787	0.2489
DELF EF	0.2883	0.2224	0.3313	0.2364	0.3831	0.2496

$EASE^R$ results are missing, because the code required too much memory on this dataset. We highlight in bold those entries where a baseline outperforms DELF.

The traditional UserKNN method is also competitive with DELF on this dataset. On Amazon Music, however, DELF EF leads to substantially better accuracy results than all baselines. As we previously observed, the test data of Amazon Music contains an anomalously high number of cold items, for which pure collaborative filtering approaches cannot learn a representation.

To assess if the performance of DELF EF is due to this uncommon test dataset, we ran additional experiments in which we removed the cold items from the test sets. This allows us to assess the true modeling capacity of the approach. The results are shown are shown in Table 18 (Amazon Music) and Table 16 (MovieLens1M). Compared to the situation where cold items are considered in the test set, the recommendation quality of DELF EF drops below many of our baselines for the Amazon Music dataset. For the MovieLens dataset, where there are no cold items, the performance does not change much as expected. Overall, these observations lead to the conclusion that DELF EF has a tendency to push never seen items to higher ranks of the recommendation list, i.e., higher than the previously seen negative samples. This might be a consequence of the DELF EF heuristic, which takes the the popularity of the items into account.

Overall, when considering only situations where cold items are not recommended, DELF was never the best-performing one. For the MovieLens1M data, iALS, EASE^R, and SLIM outperformed DELF on all measures. The UserKNN method was also consistently better in terms of the NDCG (the optimization criterion).

On the Amazon dataset, iALS was better than all other methods. Furthermore, all neighborhood models and SLIM outperformed the new methods in all but one measurement (NDCG@20). A consistent “win” for DELF method was only observed over the PureSVD method.

Looking at methodological aspects, like in other works discussed here, the authors did not optimize the number of epochs on the validation set but took the best values using the test. This is a methodological issue leading to information leakage from the test data.

A.9 Outer Product-based Neural Collaborative Filtering (ConvNCF)

Datasets. The proposed method is evaluated on two public implicit-feedback datasets, *Gowalla*[73] and *Yelp*[74]. Both datasets contain multiple implicit interactions at different timestamps for the same user-item pair. These interactions are merged by keeping only the earliest for each user-item pair.[75] Both dataset have been filtered by removing items with less than 10 interactions and users with less than 2 (Gowalla) or 10 (Yelp) interactions. The Yelp dataset has about 69k interactions after processing. The Gowalla dataset is sparser and with more interactions (1.2M). Both filtered datasets with their train/test splits have been provided by the authors.

Evaluation. Each dataset is split into training and test data according to a leave-last-out protocol. Evaluation is done by randomly selecting 999 negative items, and the algorithm has to rank these items together with the hidden item. Different methodological issues were observed, as will be discussed below. The Hit Rate and the NDCG at different list lengths are used as evaluation measures. Hyperparameter optimization is done via a validation set, except for the embedding size, which is kept constant for all methods.

Results and Discussion. We could reproduce the results using the code and the data provided by the authors. The results for the Yelp dataset are shown in Table 19. Those for the larger Gowalla dataset are given in Table 20.

For the Yelp dataset, ConvNCF is consistently outperformed by the traditional nearest-neighbor methods, RP$^3\beta$, and SLIM. The other baselines outperform ConvNCF as well in most cases. For the Gowalla case, ConvNCF is slightly more competitive. In all but one measurement, however, it is outperformed by the UserKNN method. Interestingly, the simple machine learning methods do not work better on this dataset than the simple baselines.

A number of methodological issues were observed with this article. First, based on the provided source code the number of epochs, as in other papers, was determined on the test data. Furthermore, the authors decided to set the embedding size to the constant value of 64 for all baselines. However, the embedding size is a hyperparameter to be tuned for each dataset and for each embedding-based algorithm, as different models with different objective functions or training procedures will likely require different values for it. There were also issues with the provided test splits. Negative test samples contained duplicates and partially overlapped with the train data. This results in virtually no user having the correct number of 999 unique negative items.

[73] 5http://dawenl.github.io/data/gowalla_pro.zip.

[74] https://github.com/hexiangnan/sigir16-eals.

[75] For the Gowalla dataset the authors use the first interaction appearing in the file as the *earlier* interaction.

Table 19. Experimental Results for ConvNCF for the Yelp Dataset

	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
TopPopular	0.0817	0.0538	0.1200	0.0661	0.1751	0.0799
UserKNN CF asymmetric	**0.2131**	**0.1400**	**0.3209**	**0.1747**	**0.4482**	**0.2068**
ItemKNN CF cosine	**0.2521**	**0.1686**	**0.3669**	**0.2056**	**0.4974**	**0.2385**
P$^3\alpha$	**0.2146**	**0.1395**	**0.3211**	**0.1737**	0.4442	**0.2049**
RP$^3\beta$	**0.2202**	**0.1431**	**0.3323**	**0.1793**	**0.4667**	**0.2132**
EASE^R	**0.2349**	**0.1557**	**0.3419**	**0.1902**	**0.4617**	**0.2205**
SLIM	**0.2330**	**0.1535**	**0.3475**	**0.1904**	**0.4799**	**0.2238**
PureSVD	**0.2011**	**0.1307**	0.3002	**0.1626**	0.4238	0.1938
iALS	**0.2048**	**0.1348**	**0.3080**	**0.1680**	0.4319	**0.1993**
ConvNCF	0.1947	0.1250	0.3059	0.1608	0.4446	0.1957

We highlight in bold those entries where a baseline outperforms ConvCF.

Table 20. Experimental Results for ConvNCF for the Gowalla Dataset

	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
TopPopular	0.2188	0.1652	0.2910	0.1884	0.3803	0.2110
UserKNN CF cosine	**0.7131**	**0.5879**	**0.7939**	**0.6142**	0.8532	**0.6293**
ItemKNN CF tversky	**0.7047**	**0.5864**	0.7790	**0.6105**	0.8331	**0.6244**
P$^3\alpha$	**0.6926**	**0.5703**	0.7674	**0.5948**	0.8158	**0.6071**
RP$^3\beta$	**0.6836**	**0.5525**	0.7723	**0.5814**	0.8361	**0.5976**
EASE^R	-	-	-	-	-	-
SLIM	0.6365	**0.5284**	0.7083	0.5517	0.7608	0.5651
PureSVD	0.5653	0.4482	0.6627	0.4798	0.7393	0.4993
iALS	0.6460	0.5081	0.7554	0.5436	0.8356	0.5641
ConvNCF	0.6702	0.5233	0.7799	0.5590	**0.8623**	0.5799

EASE^R results are missing, because the code required too much memory on this dataset. We highlight in bold those entries where a baseline outperforms ConvCF.

Furthermore, the users having less than 980, are only 1% for Gowalla but a remarkable 60% for Yelp, with a minimum of 938 negative items.[76]

In addition, a recent article by Ferrari Dacrema et al. [24] questioned the claim that embedding maps derived from the outer product of the embeddings are analogous to images. The article shows that embedding maps do not have a semantically relevant topology and do not share image properties, therefore the use of CNNs is not well justified. Moreover, contrary to what stated in the original paper, ConvNCF does not allow to model the embedding correlations that can be removed from the model without altering its accuracy.

[76] We report that at the time of our experiments the published version of the algorithm contained a bug that caused information leakage from the test data. Items not present in the train data were divided in two categories, those *not* present in the test data were sampled as negative items during the training of the model, while those present in the test data were not. Items unobserved in the train data should all be potential negative items to be sampled during training, regardless of the test data, otherwise test items will be advantaged over other items. This bug was later fixed on the public Github repository. In our experiments, during the training we only relied upon train data and never used any information from the test data.

A.10 Leveraging Meta-path-based Context (MCRec)

Datasets. Three datasets are used for the evaluation. The historical *MovieLens100k*[77] dataset, another small data set containing listening logs from *Last.fm,*[78] and a dataset containing user feedback from *Yelp.*[79] With almost 200k ratings, the Yelp dataset is the largest one. The available metadata includes genres for movies, artists for tracks, and city and category for businesses. Only for the MovieLens100k dataset the used data splits are publicly available. The rating information in the datasets is transformed to binary data.

Evaluation. The datasets are split into 80% training and 20% test splits. Ten percent of the training data are used for validation. The algorithm is evaluated by ranking the positive items of a user against a set of negative items randomly sampled among the items the user did not interact with. For each user, a set of exactly 50 negative items is sampled for each of the user's test items. The negative interactions in train and test data are disjoint.[80] For the MovieLens100k dataset, 23% of the users have long user profiles to the point that they cannot have the stated number of 50 negative items per each positive, because the number of items that do not appear in the user profile is too low.[81] Precision, Recall, and the NDCG are then used as evaluation measures at a cut-off length of 10. The averaged results of ten randomized runs are reported in the paper. Since the NDCG measure was implemented in an unusual way by computing the "ideal" NDCG only based on the successfully recommended items, the values reported in the paper are much higher than the ones obtained by us using a standard implementation of the NDCG. Looking at the evaluation code provided by the authors, we furthermore observed that the authors reported the best results for each metric across different epochs. Since this is inappropriate, we report the values of the metrics after the optimal number of epochs is chosen with early stopping on the validation set.

As baselines, the authors use ItemKNN, two collaborative MF methods (one based on BPR loss, the other based on cross entropy loss), two hybrid MF methods, two methods designed for metadata networks, and a number of variants of their own method. For their own method, the final hyperparameters are reported, which are the same for both datasets. For the MF and the NeuMF method, hyperparameters and configuration are taken from the original papers. Other baselines are reported to be systematically optimized on a validation set.

Results and Discussion. The meta-path information is hard-coded for the MovieLens dataset in the provided code and the preprocessed version of the data is publicly available. No code is available for the other two datasets. Furthermore, since the code for pre-processing the Yelp and Last.fm is not available, we only did an evaluation for the MovieLens dataset.[82] The results are reported in Table 21. They show that MCRec is outperformed both by the traditional neighborhood-based methods, the more complex learning-based methods and one of our simple hybrids. Our pure content-based baseline led to generally weak results (worse than TopPopular) and was also outperformed by MCRec.

[77] http://jmcauley.ucsd.edu/data/amazon/.

[78] https://www.last.fm.

[79] http://www.yelp.com/dataset-challenge.

[80] This evaluation protocol is different from more traditional protocols. Usually, both the number of positive items and negative items associated to each user is constant (e.g., 1 positive item ranked with 100 negative items), in MCRec instead different users are tested against a different number of both positive and negative samples (e.g., if the positive items are 10 the negative samples will be 500, if the positive items are 30 the negative samples will be 1,500).

[81] In particular, the longest user profile (including all available data, therefore train, validation and test interactions) that allows to sample 50 negative items for each positive (test) item is 151. However, the longest user profile in the dataset contains 737 items, which would require to sample 7,370 negative items while the dataset contains 1662 and the number of unseen items to choose from is only 925.

[82] The pre-processing code for Movielens was made available later by the authors on their GitHub repository.

Table 21. Experimental Results for MCRec for the MovieLens Dataset

	PREC@10	REC@10	NDCG@10
TopPopular	0.1907	0.1180	0.1361
UserKNN CF dice	**0.3442**	**0.2237**	**0.2692**
ItemKNN CF asymmetric	**0.3320**	**0.2171**	**0.2601**
$\mathrm{P}^3\alpha$	**0.3305**	0.2081	**0.2554**
$\mathrm{RP}^3\beta$	**0.3435**	**0.2191**	**0.2588**
EASE^R	**0.3739**	**0.2430**	**0.2905**
SLIM	**0.3770**	**0.2441**	**0.2957**
PureSVD	**0.3545**	**0.2247**	**0.2719**
iALS	**0.3596**	**0.2283**	**0.2759**
ItemKNN CBF cosine	0.0455	0.0185	0.0254
ItemKNN CFCBF cosine	**0.3398**	**0.2239**	**0.2646**
MCRec	0.3110	0.2113	0.2466

We highlight in bold those entries where a baseline outperforms MCRec.

From a methodological perspective, it is worth mentioning that the used datasets are very small, at least compared to the usual MovieLens and Netflix datasets, where we have millions of recorded interactions. We will discuss aspects of scalability in Section 5.

A.11 Collaborative Memory Network for Recommendation System (CMN)

Datasets. The method was evaluated on three public datasets. One containing data from *Epinions,*[83] a *CiteULike* dataset used in previous works (*CiteULike-a*),[84] and a dataset from *Pinterest.*[85] The last dataset is the largest one and has about 1.5M interactions. The smallest Epinions dataset contains ratings on scale from one to five, which are binarized (all non-zero ratings are set to one). The train/test splits for CiteULike and Pinterest are provided by the authors.

Evaluation. The authors use a leave-one-out methodology similar to the one used in previous works. The test set contains one positive item per user, the train set all other user-item pairs. If the user rated only one item, then this interaction is kept in the training set. To evaluate the algorithm, for each user 100 unobserved (negative) items are sampled and ranked together with the positive item. The Hit Rate and the NDCG at a cut-off length of 10 are used as evaluation metrics.

As baselines, the authors include both simple, non-neural, and neural methods in there experiments: ItemKNN, BPR matrix-factorization, SVD++, two variants of NCF [30], and the Collaborative Denoising Auto Encoder.

Hyperparameters are tuned on a validation set. Details for the hyperparameter optimization process are provided for the proposed model, but not for the baselines. The number of training epochs is not mentioned in the paper. In our evaluation, we applied early stopping as described previously.

Results and Discussion. We could reproduce the results of the paper using the data splits provided by the authors for two datasets, CiteULike and Pinterest. Since the split used for Epinions was not made available, we had to re-create it based on the information provided in the paper, in that case, however, we could not reproduce the original results. In Table 22, we show the results for

[83] http://www.trustlet.org/downloaded_epinions.html.

[84] See Section A.1.

[85] https://sites.google.com/site/xueatalphabeta/academic-projects.

Table 22. Experimental Results for CMN for the Pinterest Dataset

	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1665	0.1064	0.2740	0.1409
UserKNN CF asymmetric	0.7005	**0.5037**	0.8630	**0.5567**
ItemKNN CF cosine	**0.7132**	**0.5116**	**0.8781**	**0.5653**
$P^3\alpha$	0.6990	**0.5034**	0.8596	**0.5559**
$RP^3\beta$	**0.7147**	**0.5150**	**0.8772**	**0.5680**
$EASE^R$	**0.7072**	**0.5129**	0.8567	**0.5617**
SLIM	**0.7084**	**0.5107**	**0.8683**	**0.5628**
PureSVD	0.6619	0.4721	0.8146	0.5219
iALS	**0.7219**	**0.5175**	**0.8677**	**0.5652**
CMN	0.7013	0.5005	0.8674	0.5547

We highlight in bold those entries where a baseline outperforms CMN.

the Pinterest dataset, the largest one. For the CMN method, we report the results for the *CMN-3* variant, which led to the best results.

The results show that CMN led to competitive results, but is actually slightly outperformed on all measures by algorithms from all families, including traditional nearest-neighbor methods. On the CiteULike dataset, the main observations are the same. On this smaller dataset, however, the performance of CMN is often much lower than the one achieved by nearest-neighbor methods and machine learning techniques.

A quite surprising result is found for the Epinions dataset. Here, the trivial TopPopular method consistently led to the best values across all measurements. The difference to all other methods is often huge. This indicates that the popularity distribution of the items in this dataset is very skewed, which makes it difficult to make personalized recommendations that are better in terms of information retrieval measures than the TopPopular method. The detailed results are available in the online material (Section 3).

A.12 Spectral Collaborative Filtering (SpectralCF)

Datasets. Three public datasets are used for the evaluation, *MovieLens1M,*[86] another movie rating dataset (*HetRec*),[87] and an *Amazon* dataset (*InstantVideo*).[88] Explicit ratings are binarized. Further pre-processing is applied to, e.g., remove users associated to less than 5 interactions. After pre-processing, the MovieLens1M dataset is shrinked to one fifth of the original size (226k interactions). The other datasets are even smaller (71k and 22k interactions, respectively). The data split for the MovieLens1M dataset is provided online by the authors. Due to an apparent problem on how the splits were generated—see our discussions below—we created our own splits based on the information provided in the paper.

Evaluation. Two evaluation scenarios are tested, a regular one and a cold-start setup. We evaluated both scenarios. For the main scenario, 80% of the interactions of each user are randomly put into the training set and the rest is used for evaluation. The random process is repeated five times and averaged results are reported. Recall and MAP at different list lengths are used as metrics for this scenario.

[86] https://grouplens.org/datasets/movielens/.
[87] https://grouplens.org/datasets/hetrec-2011/.
[88] http://jmcauley.ucsd.edu/data/amazon/.

Table 23. Experimental Results for SpectralCF for the MovieLens Dataset

	REC@20	MAP@20	REC@40	MAP@40	REC@60	MAP@60	REC@80	MAP@80	REC@100	MAP@100
TopPopular	**0.1892**	**0.0584**	**0.2788**	**0.0636**	**0.3356**	**0.0666**	**0.3834**	**0.0687**	0.4226	**0.0702**
UserKNN CF jaccard	**0.3001**	**0.1201**	**0.4134**	**0.1285**	**0.4901**	**0.1335**	**0.5457**	**0.1367**	**0.5884**	**0.1388**
ItemKNN CF asymmetric	**0.2876**	**0.1134**	**0.4000**	**0.1213**	**0.4768**	**0.1263**	**0.5367**	**0.1295**	**0.5820**	**0.1317**
P3alpha	**0.2939**	**0.1141**	**0.4150**	**0.1233**	**0.4900**	**0.1285**	**0.5463**	**0.1318**	**0.5903**	**0.1342**
RP3beta	**0.2737**	**0.1044**	**0.3879**	**0.1124**	**0.4664**	**0.1173**	**0.5234**	**0.1206**	**0.5726**	**0.1230**
EASE^R	**0.3085**	**0.1249**	**0.4255**	**0.1340**	**0.4986**	**0.1391**	**0.5559**	**0.1425**	**0.6010**	**0.1448**
SLIM	**0.3069**	**0.1265**	**0.4246**	**0.1356**	**0.5010**	**0.1410**	**0.5564**	**0.1443**	**0.6001**	**0.1466**
PureSVD	**0.2595**	**0.1008**	**0.3638**	**0.1083**	**0.4378**	**0.1131**	**0.4913**	**0.1161**	**0.5347**	**0.1182**
iALS	**0.3033**	**0.1183**	**0.4201**	**0.1273**	**0.4933**	**0.1326**	**0.5493**	**0.1360**	**0.5925**	**0.1383**
SpectralCF	0.1813	0.0533	0.2643	0.0581	0.3274	0.0613	0.3823	0.0635	0.4261	0.0651

We highlight in bold those entries where a baseline outperforms SpectralCF.

In the cold-start scenario, the training set is built with different degrees of sparsity by varying the number P of interactions associated with each user, where P is varied from one to five. The remaining items associated with each users are used as test set. Recall@20 and MAP@20 are used in this scenario for evaluation.

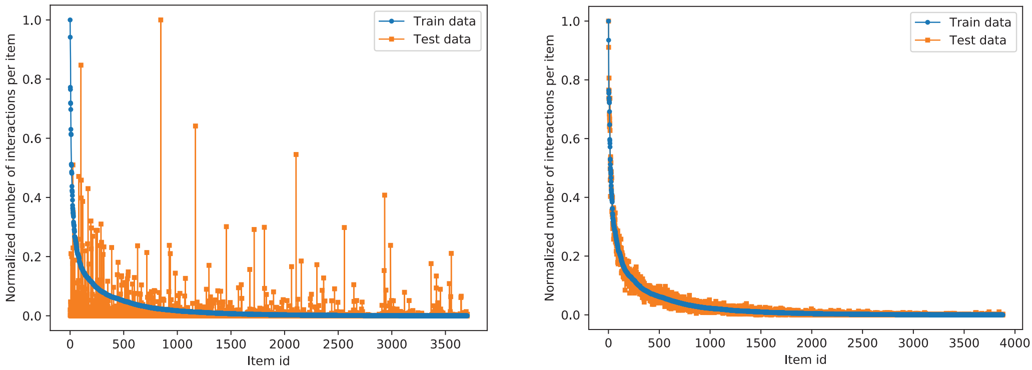
As baselines, the authors consider ItemKNN, BPR matrix factorization, iALS, NCF [30] and two graph-based methods, GNMF and GCMC, originally designed for explicit feedback datasets. The hyperparameters are tuned on a validation set and the parameter ranges are reported in the paper. For NCF the authors stated that they used the configuration reported in the original paper.

Results and Discussion. We reproduced the results obtained by the authors based on the provided code and following the information in the paper. Our first set of experiments showed that on HetRec and Amazon Video datasets the performance of SpectralCF was relatively weak, whereas it performed very well and better than all baselines when using the provided data splits for the MovieLens dataset. We therefore investigated the provided data splits and found that these splits were unlikely the result of the splitting procedure described in the paper, see the discussions below. For this reason, we created new data splits ourselves following the described procedure.

The results obtained for the MovieLens1M dataset are shown in Table 23. The results show that SpectralCF is outperformed by all baselines in our comparison often exhibiting a recommendation quality equal to the TopPopular method. The observations for the other datasets are similar, the full results are available in the online material (Section 3).

To illustrate the data splitting problem, we compared the popularity distribution of the items in the training and the test split in the provided data, see Figure 3(a). The figure plots the normalized popularity (i.e., the popularity of an item over the popularity of the most popular item) for both training and test items. Items are ranked based on their popularity in the training set, with the most popular training items being on the left. In case of a true random split, the normalized popularity values of the items in the training and the test split should be relatively close. However, the figure shows the split provided by the authors has some major deviations. Figure 3(b) shows the popularity distributions of our random split, which are almost identical between training and test sets.

Besides the visual analysis, we also computed numerical statistics like the Gini index and Shannon entropy. In a true random split, the Gini index should be close to the value obtained on the unsplitted data (in this case 0.78) for both the training and test splits. However, the Gini index



(a) Normalized popularity distributions of the train and test splits provided by the original authors.

(b) Normalized popularity distributions of the train and test splits generated by us.

Fig. 3. Normalized popularity distributions of the train and test splits for SpectralCF, the value 1 corresponds to the most popular item in that split. For a random split, as can be seen in panel (b), the normalized values of both splits are, on average, similar. In the split provided by the original authors, however, as can be seen in panel (a), train and test data have quite different distributions.

of the provided test split is much higher (0.92). This indicates that the provided test split has a much higher popularity bias than we would expect. A similar consideration applies for the Shannon entropy: the entropy of the original dataset is close to 10 and is similar to the entropy of our random train/test split. However, the provided test split has a lower entropy (8.5), i.e., it is easier to predict.

Regarding other methodological aspects, we found that the authors only report one set of hyperparameters, whereas one would expect hyperparameter settings for each dataset. In our evaluation, we therefore optimized the hyperparameters for all baselines and all datasets individually.

B HYPERPARAMETER RANGES

Table 24. Hyperparameter List, Value Ranges, and Distributions for the Baselines Reported in This Paper

Algorithm	Hyperparameter	Range	Type	Distribution
UserKNN, ItemKNN cosine	topK	5–1,000	Integer	uniform
	shrink	0–1,000	Integer	uniform
	normalize[a]	True, False	Categorical	
	feature weighting	none, TF-IDF, BM25	Categorical	
UserKNN, ItemKNN dice	topK	5–1,000	Integer	uniform
	shrink	0–1,000	Integer	uniform
	normalize[a]	True, False	Categorical	
UserKNN, ItemKNN jaccard	topK	5–1,000	Integer	uniform
	shrink	0–1,000	Integer	uniform
	normalize[a]	True, False	Categorical	
UserKNN, ItemKNN asymmetric	topK	5–1,000	Integer	uniform
	shrink	0–1,000	Integer	uniform
	normalize[a]	True	Categorical	
	asymmetric alpha	0–2	Real	uniform
	feature weighting	none, TF-IDF, BM25	Categorical	
UserKNN, ItemKNN tversky	topK	5–1,000	Integer	uniform
	shrink	0–1,000	Integer	uniform
	normalize[a]	True	Categorical	
	tversky alpha	0–2	Real	uniform
	tversky beta	0–2	Real	uniform
P3alpha	topK	5–1,000	Integer	uniform
	alpha	0–2	Real	uniform
	normalize similarity[b]	True, False	Categorical	
RP3beta	topK	5–1,000	Integer	uniform
	alpha	0–2	Real	uniform
	beta	0–2	Real	uniform
	normalize similarity[b]	True, False	Categorical	
SLIMElasticNet	topK	5–1,000	Integer	uniform
	l1 ratio	10^{-5}– 10^{0}	Real	log-uniform
	alpha	10^{-3}– 10^{0}	Real	uniform
PureSVD	num factors	1–350	Integer	uniform
iALS	num factors	1–200[c]	Integer	uniform
	epochs	1–500	Integer	early-stopping
	confidence scaling	linear, log	Categorical	
	alpha	10^{-3}– $5 \cdot 10^{+1}$[d]	Real	log-uniform
	epsilon	10^{-3}– 10^{+1}[d]	Real	log-uniform
	reg	10^{-5}– 10^{-2}	Real	log-uniform
EASE^R	l2 norm	10^{0}– 10^{+7}	Real	log-uniform

[a] The *normalize* hyperparameter in KNNs refers to the use of the denominator when computing the similarity.
[b] The *normalize similarity* hyperparameter refers to applying L1 regularisation on the rows of the similarity matrix.
[c] The number of factors is lower than PureSVD due to iALS being slower.
[d] The maximum value of this hyperparameter had been suggested in the article proposing the algorithm.

REFERENCES

[1] Fabio Aiolli. 2013. Efficient top-n recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys'13)*. 273–280.

[2] Sebastiano Antenucci, Simone Boglio, Emanuele Chioso, Ervin Dervishaj, Shuwen Kang, Tommaso Scarlatti, and Maurizio 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*. ACM, 4:1–4:6. DOI: https://doi.org/10.1145/3267471.3267475

[3] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. 2009. Improvements that don't add up: Adhoc retrieval results since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09)*. 601–610.

[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 of the 19th International Conference on Theory and Practice of Digital Libraries (TPDL'15)*. 153–168.

[5] Robert M. Bell and Yehuda Koren. 2007. Improved neighborhood-based collaborative filtering. In *Proceedings of the KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*. 7–14.

[6] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* 13 (Feb.2012), 281–305.

[7] Shlomo Berkovsky, Ivan Cantador, and Domonkos Tikk (Eds.). 2019. *Collaborative Recommendations—Algorithms, Practical Challenges and Applications*. World Scientific.

[8] Homanga Bharadhwaj, Homin Park, and Brian Y. Lim. 2018. RecGAN: Recurrent generative adversarial networks for recommendation systems. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys'18)*. 372–376. DOI: https://doi.org/10.1145/3240323.3240383

[9] Daniel Billsus and Michael J. Pazzani. 1998. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*. 46–54.

[10] John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*. 43–52.

[11] Rocío Cañamares, Pablo Castells, and Alistair Moffat. 2020. Offline evaluation options for recommender systems. *Info. Retriev. J.* 23, 4 (2020), 387–410.

[12] 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 of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'17)*. 335–344.

[13] Weiyu Cheng, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. 2018. DELF: A dual-embedding based deep latent factor model for recommendation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 3329–3335.

[14] Andy Cockburn, Pierre Dragicevic, Lonni Besançon, and Carl Gutwin. 2020. Threats of a replication crisis in empirical computer science. *Commun. ACM* 63, 8 (2020), 70–79.

[15] Christian Collberg and Todd A. Proebsting. 2016. Repeatability in computer systems research. *Commun. ACM* 59, 3 (2016), 62–69.

[16] Colin Cooper, Sang Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random walks in recommender systems: Exact computation and simulations. In *Proceedings of the 23rd International Conference on World Wide Web (WWW'14)*. 811–816.

[17] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. 2012. Investigating the persuasion potential of recommender systems from a quality perspective: An empirical study. *Trans. Interact. Intell. Syst.* 2, 2 (2012), 1–41.

[18] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys'10)*. 39–46.

[19] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2020. Methodological issues in recommender systems research (extended abstract). In *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI 2020*. ijcai.org, 4706–4710. DOI: https://doi.org/10.24963/ijcai.2020/650

[20] Lee R. Dice. 1945. Measures of the amount of ecologic association between species. *Ecology* 26, 3 (1945), 297–302. DOI: https://doi.org/10.2307/1932409 arXiv:https://esajournals.onlinelibrary.wiley.com/doi/pdf/10.2307/1932409

[21] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'18)*. 515–524.

[22] 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 of the 24th International Conference on World Wide Web (WWW'15)*. 278–288.

[23] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys'19)*. 101–109. DOI: https://doi.org/10.1145/3298689.3347058 Source: https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation.

[24] Maurizio Ferrari Dacrema, Federico Parroni, Paolo Cremonesi, and Dietmar Jannach. 2020. Critically examining the claimed value of convolutions over user-item embedding maps for recommender systems. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*. DOI: https://doi.org/10.1145/3340531.3411901

[25] Juliana Freire, Norbert Fuhr, and Andreas Rauber. 2016. Reproducibility of data-oriented experiments in e-science (dagstuhl seminar 16041). *Dagstuhl Rep.* 6, 1 (2016), 108–159.

[26] Antonino Freno, Martin Saveski, Rodolphe Jenatton, and Cédric Archambeau. 2015. One-pass ranking models for low-latency product recommendations. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15)*. 1789–1798. DOI: https://doi.org/10.1145/2783258.2788579

[27] 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 of the 18th ACM Conference on Recommender Systems (RecSys'14)*. 169–176.

[28] Asela Gunawardana and Guy Shani. 2015. Evaluating recommender systems. In *Recommender Systems Handbook*. Springer, 265–308.

[29] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer product-based neural collaborative filtering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 2227–2233.

[30] 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.

[31] José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. 2014. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 918–926. Retrieved from http://papers.nips.cc/paper/5324-predictive-entropy-search-for-efficient-global-optimization-of-black-box-functions.pdf.

[32] 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 of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18)*. 1531–1540.

[33] 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'08)*, Vol. 8. 263–272.

[34] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'16)*. 233–240. DOI: https://doi.org/10.1145/2959100.2959165

[35] Doudou LaLoudouana and Mambobo Bonouliqui Tarare.2002. Data set selection. Retrieved from http://www.jmlg.org/papers/laloudouana03.pdf.

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

[37] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'17)*. 305–314.

[38] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 27th International Conference on World Wide Web (WWW'18)*. 689–698.

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

[40] Jimmy Lin. 2019. The neural hype, justified! A recantation. *SIGIR Forum* 53, 2 (2019), 88–93.

[41] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Comput.* 7, 1 (2003), 76–80.

[42] Zachary C. Lipton and Jacob Steinhardt. 2019. Troubling trends in machine learning scholarship. *Queue* 17, 1, Article Pages 80 (Feb. 2019), 33 pages. DOI: https://doi.org/10.1145/3317287.3328534

[43] 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.

[44] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User-Model. User-Adapt. Interact.* 28, 4–5 (2018), 331–390.

[45] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Performance comparison of neural and non-neural approaches to session-based recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys'19)*. 462–466.

[46] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2018. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS One* 13, 3 (2018).

[47] Andrii Maksai, Florent Garcin, and Boi Faltings. 2015. Predicting online performance of news recommender systems through richer evaluation metrics. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys'15)*. 179–186.

[48] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2018. A contextual attention recurrent architecture for context-aware venue recommendation. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'18)*. 555–564. DOI: https://doi.org/10.1145/3209978.3210042

[49] Sean M. McNee, John Riedl, and Joseph A. Konstan. 2006. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *Proceedings of the CHI Extended Abstracts on Human Factors in Computing Systems (CHI EA'06)*. 1097–1101. DOI: https://doi.org/10.1145/1125451.1125659

[50] Bamshad Mobasher, Xin Jin, and Yanzan Zhou. 2004. Semantically enhanced collaborative filtering on the web. In *Web Mining: From Web to Semantic Web*, Bettina Berendt, Andreas Hotho, Dunja Mladenič, Maarten van Someren, Myra Spiliopoulou, and Gerd Stumme (Eds.). Springer Berlin, 57–76.

[51] ThaiBinh Nguyen and Atsuhiro Takasu. 2018. NPE: Neural personalized embedding for collaborative filtering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 1583–1589. Retrieved from http://dl.acm.org/citation.cfm?id=3304415.3304640.

[52] Xia Ning and George Karypis. 2011. SLIM: Sparse linear methods for top-n recommender systems. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM'11)*. 497–506.

[53] Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of the KDD Cup and Workshop*. 39–42.

[54] 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.

[55] Ali Mustafa Qamar, Éric Gaussier, Jean-Pierre Chevallet, and Joo-Hwee Lim. 2008. Similarity learning for nearest-neighbor classification. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*. 983–988. DOI: https://doi.org/10.1109/ICDM.2008.81

[56] 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)*.

[57] Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: A study on recommender systems. Retrieved from http://arxiv.org/abs/1905.01395.

[58] 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 ACM Conference on Computer-Supported Cooperative Work (CSCW'94)*. 175–186.

[59] Marco Rossetti, Fabio Stella, and Markus Zanker. 2016. Contrasting offline and online results when evaluating recommendation algorithms. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'16)*. 31–34.

[60] Noveen Sachdeva, Kartik Gupta, and Vikram Pudi. 2018. Attentive neural architecture incorporating song features for music recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys'18)*. 417–421. DOI: https://doi.org/10.1145/3240323.3240397

[61] 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.

[62] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Proceedings of the 5th International Conference on Computer and Information Science (ICIS'02)*, Vol. 1. Citeseer, 27–8.

[63] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *Proceedings of the 28th International Conference on World Wide Web (WWW'19) (TheWebConf'19)*. 3251–3257.

[64] Victoria Stodden. 2010. The scientific method in practice: Reproducibility in the computational sciences. *MIT Sloan Research Paper, No. 4773-10* (2010).

[65] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys'18)*. 297–305. DOI: https://doi.org/10.1145/3240323.3240361

[66] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the 27th International Conference on World Wide Web (WWW'18)*. 729–739.

[67] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18)*. 2309–2318.

[68] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D convolutional networks for session-based recommendation with content features. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys'17)*. 138–146. DOI: https://doi.org/10.1145/3109859.3109900

[69] Amos Tversky. 1977. Features of similarity. *Psychol. Rev.* 84, 4 (1977), 327–352.

[70] Patrick Vandewalle, Jelena Kovacevic, and Martin Vetterli. 2009. Reproducible research in signal processing. *IEEE Signal Process. Mag.* 26, 3 (2009), 37–47.

[71] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'16)*. 225–232. DOI:https://doi.org/10.1145/2959100.2959160

[72] Kiri Wagstaff. 2012. Machine learning that matters. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*. 529–536.

[73] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15)*. 1235–1244.

[74] Jun Wang, Stephen Robertson, Arjen P. de Vries, and Marcel J. T. Reinders. 2008. Probabilistic relevance ranking for collaborative filtering. *Info. Retriev.* 11, 6 (2008), 477–497.

[75] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM'16)*. 153–162.

[76] Zhenghua Xu, Thomas Lukasiewicz, Cheng Chen, Yishu Miao, and Xiangwu Meng. 2017. Tag-aware personalized recommendation using a hybrid deep model. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. 3196–3202. DOI:https://doi.org/10.24963/ijcai.2017/446

[77] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 3203–3209. DOI:https://doi.org/10.24963/ijcai.2017/447

[78] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. Critically examining the neural hype: Weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 1129–1132.

[79] Quangui Zhang, Longbing Cao, Chengzhang Zhu, Zhiqiang Li, and Jinguang Sun. 2018. CoupledCF: Learning explicit and implicit user-item couplings in recommendation for deep collaborative filtering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 3662–3668. DOI:https://doi.org/10.24963/ijcai.2018/509

[80] Shuai Zhang, Lina Yao, Aixin Sun, Sen Wang, Guodong Long, and Manqing Dong. 2018. NeuRec: On nonlinear transformation for personalized ranking. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 3669–3675.

[81] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys'18)*. 311–319.

Received November 2019; revised August 2020; accepted November 2020