Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison

ZHU SUN, Macquarie University, Australia
DI YU, Shanghai University of Finance and Economics, China
HUI FANG*, Shanghai University of Finance and Economics, China
JIE YANG, Delft University of Technology, The Netherlands
XINGHUA QU, Nanyang Technological University, Singapore
JIE ZHANG, Nanyang Technological University, Singapore
CONG GENG, Shanghai University of Finance and Economics, China

With tremendous amount of recommendation algorithms proposed every year, one critical issue has attracted a considerable amount of attention: there are no effective benchmarks for evaluation, which leads to two major concerns, i.e., unreproducible evaluation and unfair comparison. This paper aims to conduct rigorous (i.e., reproducible and fair) evaluation for implicit-feedback based top-N recommendation algorithms. We first systematically review 85 recommendation papers published at eight top-tier conferences (e.g., RecSys, SIGIR) to summarize important evaluation factors, e.g., data splitting and parameter tuning strategies, etc. Through a holistic empirical study, the impacts of different factors on recommendation performance are then analyzed in-depth. Following that, we create benchmarks with standardized procedures and provide the performance of seven well-tuned state-of-the-arts across six metrics on six widely-used datasets as a reference for later study. Additionally, we release a user-friendly Python toolkit, which differs from existing ones in addressing the broad scope of rigorous evaluation for recommendation. Overall, our work sheds light on the issues in recommendation evaluation and lays the foundation for further investigation. Our code and datasets are available at GitHub (https://github.com/AmazingDD/daisyRec).

CCS Concepts: • **Information systems** → **Collaborative filtering**; **Personalization**.

Additional Key Words and Phrases: Recommender Systems, Reproducible Evaluation, Fair Comparison, Benchmarks

ACM Reference Format:
Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 21–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3383313.3412489

1 INTRODUCTION

Recent decades have witnessed the great prosperity of recommender systems in both academia and industry, thanks to their effectiveness in addressing the information overload problem [31]. Aiming to provide personalized services to customers, recommendation algorithms have been widely studied and applied in various domains, ranging from

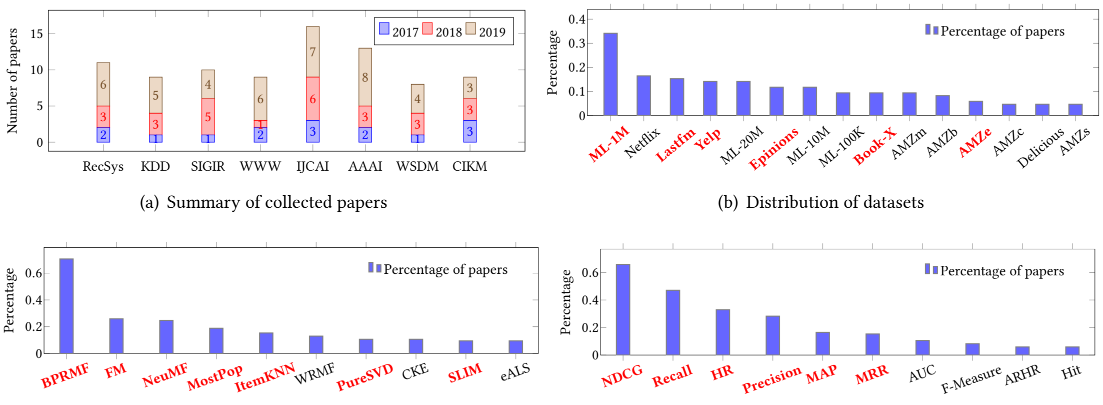
[*] Hui Fang is the corresponding author. Email address: fang.hui@mail.shufe.edu.cn

e-commerce (e.g., Amazon, Tmall) to location-based social networks (e.g., Foursquare, Yelp) and multi-media (e.g., Netflix, Spotify). Existing recommendation algorithms are dominated by three types of underlying techniques: memory-based methods (MMs) [7, 29], latent factor-based methods (LFMs) [31], and representation learning-based methods (RLMs), including item embedding-based methods [2], and deep learning-based methods (DLMs) [14, 45].

With tremendous amount of recommendation algorithms being proposed, one critical issue has attracted much attention from researchers in the community: there are no effective benchmarks for evaluation. It, consequently, leads to two major concerns, namely unreproducible evaluation and unfair comparison. These problems were first pointed out by [27, 28]. A recent study [25] shows that the results of baseline algorithms reported in numerous publications over the past five years are suboptimal. With a careful setup, the baselines can outperform most of the newly proposed method. This is consistent with another latest study [7], which discovers that the recently proposed DLMs [14] can be defeated by simpler baselines, such as MostPop and ItemKNN [29] with fine-tuned parameters. These findings have triggered heated discussions on the evaluation of recommendation methods.

Different from other domains such as computer vision, where mature benchmarks [8] are available to fairly evaluate the newly proposed approaches, benchmarking recommendation is more challenging in two aspects: (1) there exists a lot of datasets from different platforms in each application domain. For example, widely used datasets in the movie domain includes MovieLens (ML), Netflix and Amazon-Movie, etc. Even for the same dataset, it may have various versions with sizes covering different durations, e.g., ML-100K/1M/10M/20M/25M/Latest. It is common that researchers choose different datasets heuristically, and only report results on the selected datasets; (2) there are different data-processing strategies, data splitting methods, evaluation metrics and parameter settings, etc. For instance, some studies adopt 5-filter setting [41] to filter out users and items with less than 5 ratings, while others adopt 10-filter setting [37, 38]. In terms of data splitting methods, some utilize leave-one-out [14], while others leverage split-by-ratio (e.g., training: validation: test = 80%: 10%: 10%) [37]. Most importantly, the majority of papers do not report details on data processing and parameter settings, leading to inconsistent results in reproduction by different researchers.

These issues motivate us to investigate evaluation rigorousness (i.e., reproducibility and fairness) in recommendation. We choose to focus on the top-N recommendation task, one of the most prominent tasks in recommendation, and leave other tasks (e.g., temporal, session, location, group and cross domain aware recommendation) for future exploration. To achieve this, **(1)** we first systematically review 85 papers related to implicit feedback based top-N recommendation published in the recent three years (2017-2019) on eight top-tier conferences as representatives, including RecSys, KDD, SIGIR, WWW, IJCAI, AAAI, WSDM and CIKM (most important venues that accept high-quality recommendation papers). From those papers, we summarize essential factors related to evaluation, including utilized datasets, data pre-processing strategies, comparison baselines, loss function designs, negative sampling strategies, data splitting methods, evaluation metrics, and parameter tuning strategies. **(2)** Through a holistic empirical study, we then comprehensively analyze the influence of different factors on recommendation performance. Our results further confirm the findings obtained in the recent study [7, 25]. Besides, several interesting observations are noted, for example, (a) the recommendation performance does not necessarily improve with denser datasets; (b) the objective function with pair-wise log loss generally outperforms other types of objective functions; (c) uniform sampler, though simple, performs better than the popularity based sampler; and (d) the best hyper-parameter settings for one specific metric does not necessarily guarantee optimums w.r.t. other metrics. **(3)** Based on the empirical results, we create benchmarks where we propose the standardized procedures to improve the reproducibility and fairness of evaluation. Meanwhile, the performance of seven well-tuned state-of-the-arts on six widely-used datasets across six metrics is provided as a reference for later study. **(4)** Furthermore, we have released a user-friendly Python toolkit for rigorous evaluation. We point out that



(a) Summary of collected papers

(b) Distribution of datasets

(c) Distribution of baselines

(d) Distribution of evaluation metrics

Fig. 1. (a) shows the summary of the collected papers; (b) depicts the popularity of top-15 utilized datasets, where 'ML, Book-X, AMZm/b/e/c/s' are MovieLens, Book-Crossing and Amazon Music/Book/Electronic/Clothing/Sports, respectively. The six selected datasets are highlighted in red; (c) shows the popularity of top-10 comparison baselines, where the selected seven baselines are highlighted in red.; and (d) illustrates the popularity of evaluation metrics, where the six selected metrics are highlighted in red.

although there are several existing libraries (e.g., LibRec [11] and DeepRec [44]) in recommendation, they mainly aim to reproduce the logic of various state-of-the-arts. They seldom consider reproducibility from the angle of performing rigorous evaluation in recommendation. To sum up, our work sheds lights on the urgent issues of rigorousness in the evaluation for recommendation, and lays foundation for further investigation on this topic.

2 PAPER COLLECTION AND ANALYSIS

2.1 Paper Collection

To achieve rigorous evaluation for recommendation, we first conduct a comprehensive review over papers published in the recent three years (2017-2019) on eight top-tier conferences, namely, RecSys, KDD, SIGIR, WWW, IJCAI, AAAI, WSDM and CIKM. As a starting point, we mainly focus on recommendation methods for implicit feedback based top-N recommendation, which is one of the most important tasks in recommendation. Other tasks (e.g., session-aware recommendation) are left for future exploration. We first search the accepted full paper lists ($8 * 3 = 24$) for the eight conferences in the three years. Given our interest and the 24 lists, we consider papers with titles containing the keywords 'recommend$*$' or 'collaborative filtering'. After that, we manually select papers that adopt ranking metrics (e.g., Precision, Recall) to evaluate the performance of recommendation. In the end, we obtain a collection of 85 papers and summarize their distribution over the conferences and publication years in Figure 1(a). Due to space limitation, details of these papers are listed in the Additional Material of our GitHub repository. Based on this collection, we conduct a systematic analysis to investigate the essential factors related to evaluation, including the *utilized datasets*, *data pre-processing*, *comparison baselines*, *loss function designs*, *negative sampling strategies*, *data splitting methods*, *evaluation metrics*, and *parameter tuning strategies*, as detailed below.

2.2 Paper Analysis

2.2.1 Utilized Datasets. Analyzing the collected papers, we find two major issues of the utilized datasets: (1) domain diversity, i.e., there are plenty of datasets within and across various domains, as shown in the Additional Material of our GitHub repository; (2) version diversity, i.e., many datasets, although with the same name, may have different

Table 1. Statistics of datasets.

Dataset		ML-1M	Lastfm	Yelp	Epinions	Book-X	AMZe
origin	#User	6,038	1,892	1,326,101	22,164	105,283	4,201,696
	#Item	3,533	17,632	174,567	296,277	340,556	476,002
	#Record	575,281	92,834	5,261,669	922,267	1,149,780	7,824,482
	Density	2.697e-2	2.783e-3	2.273e-5	1.404e-4	3.207e-5	3.912e-6
5-filter	#User	6,034	1,874	227,109	21,995	22,072	253,994
	#Item	3,125	2,828	123,985	31,678	43,748	145,199
	#Record	574,376	71,411	3,419,587	550,117	623,405	2,109,869
	Density	3.046e-2	1.348e-2	1.214e-4	7.895e-4	6.456e-4	5.721e-5
10-filter	#User	5,950	1,867	96,168	21,111	12,720	63,161
	#Item	2,811	1,530	80,351	14,030	18,318	85,930
	#Record	571,549	62,984	2,458,153	434,162	443,196	949,416
	Density	3.412e-2	2.205e-2	3.181e-4	1.466e-3	1.902e-3	1.749e-4
Timestamp		√	×	√	√	×	√

versions. For example, we find more than three versions for Yelp, as it has been updated for different rounds of the hosting challenge. Overall, there are 65 different datasets used in the 85 papers (different versions of a dataset counted only once). Figure 1(b) shows the popularity of the top-15 datasets , measured by the percentage of relevant papers: around 90% of the 85 papers adopt at least one of the 15 datasets. Considering dataset popularity and domain coverage, we select six datasets for our study as highlighted in red in Figure 1(b): **ML-1M** (Movie), **Lastfm** (Music), **Yelp** (LBSNs), **Epinions** (SNs), **Book-X** (Book) and **AMZe** (Consumable). The selected datasets cover 67% papers of the collection.

To ease the version diversity issue, we conduct a careful selection by considering the authority and information richness of data sources, which could benefit the study on diverse recommendation models. Specifically, we use MovieLens-1M (ML-1M) released by GroupLens (grouplens.org/datasets/movielens/); Lastfm was released by the 2nd international workshop HetRec 2011 (ir.ii.uam.es/hetrec2011/); Yelp was created by Kaggle in 2018 (www.kaggle.com/yelp-dataset/yelp-dataset); Epinions was crawled by [33] containing timestamp and item category information; Book-Crossing (Book-X) [47] was collected by Cai-Nicolas Ziegler from the Book-Crossing community (grouplens.org/datasets/book-crossing/); Amazon Electronic (AMZe) was released by Julian McAuley (jmcauley.ucsd.edu/data/amazon/links.html). The statistics of all datasets are listed in Table 1.

2.2.2 Data Pre-Processing. As we focus on implicit feedback, datasets with explicit feedback (e.g., ratings or counts) are binarized into implicit data. Let $u \in \mathcal{U}, i \in \mathcal{I}$ denote user u and item i; $\mathcal{U}, \mathcal{I}$ are user and item sets; and $r_{ui} \in \mathcal{R}$ is the binary feedback of user u over item i. For each user, we convert all her explicit feedback with no less than a threshold (denoted by t) into positive feedback ($r_{ui} = 1$); otherwise, negative feedback ($r_{ui} = 0$). Regarding the threshold t, different papers may have different settings (e.g., $t = 1/2/3/4$). Following majority studies [34, 36, 39], we set $t = 4$ for ML-1M, and $t = 1$ for the rest datasets.

Generally, the original recommendation datasets are extremely sparse, where most users only interact with a small number of items, e.g., less than five. In evaluation, the original datasets are usually pre-processed to filter out the extremely inactive users and items. By analyzing the paper collection, we have found out that around 50% of papers adopt pre-processing strategies; while 27% of papers utilize the original datasets; and 23% of papers do not report details on data processing. Among the papers adopting pre-processing strategies, more than 60% of them utilize 5- or 10-filter setting on the datasets, i.e., by filtering out users and items with less than 5 or 10 interactions, respectively. While, others adopt, such as 1-, 2-, 3-, 4-, 20- or 30-filter settings, on the datasets. To measure the performance and robustness of methods across different data sparsity, we take, besides original datasets, the two most common settings (i.e., **5-** and **10-filter**) on all selected datasets. Detailed statistics of the datasets are summarized in Table 1. Note that $K-$fiter is different from $K-$core: the former means that users and items are only filtered with less than K interactions in one pass; by contrast, the latter indicates a recursive filter until all users and items have at least K interactions.

2.2.3 Comparison Baselines. As observed from the collected papers, the compared baselines vary a lot in different papers. We show the top-10 widely-compared baselines in these papers in Figure 1(c), covering 93% of papers in total, that is, 93% of the papers consider at least one of the 10 baselines. The baselines fall into three broad categories, (1) memory-based methods (MMs): MostPop, ItemKNN [29]; (2) latent factor-based methods (LFMs): BPRMF [26], FM [24], WRMF [17], PureSVD [6], SLIM [22] and eALS [15]; and (3) representation learning-based methods (RLMs): NeuMF [14] and CKE [43]. For our study, we take 7 baselines into account, as highlighted in red in Figure 1(c).

In particular, two MMs are considered: **MostPop** is a non-personalized method and recommends most popular items to all users; **ItemKNN** is a K-nearest neighborhood based method recommending items based on item similarity. We adapt it for implicit feedback data following [17], and adopt cosine similarity. In terms of the LFMs, we select **BPRMF** as the representative of matrix factorization method, and leave WRMF and eALS for future exploration; **BPRFM** (factorization machine) considers the second-order feature interactions between inputs. We train it by optimizing the BPR loss [26]; **PureSVD** directly performs conventional singular value decomposition on the user-item implicit interaction matrix, where all the unobserved entries are set as 0; **SLIM** [22] learns a sparse item-item similarity matrix by minimizing a constrained reconstruction square loss. Regarding to the RLMs, **NeuMF** [14] is considered, which is a state-of-the-art neural network method. CKE is left for future study, as it involves textual and visual information in addition to the user-item interaction data. Besides, more advanced deep learning baselines, e.g., NeuFM [13], VAE [19], CDAE [40] are also left for future exploration.

2.2.4 Objective Function. Two types of objective functions are widely utilized: **point-wise** (49% of the collected papers) and **pair-wise** (42% of the collected papers). The former only relies on the accuracy of the prediction of individual preferences; whilst the latter approximates ranking loss by considering the relative order of the predictions for pairs of items. Regardless of which one is deployed, it is critical to properly exploit unobserved feedback within the model, as merely considering the observed feedback fails to account for the fact that feedback is not missing at random, thus being not suitable for top-N recommenders [40].

Let $\mathcal{L}$ denote the objective function of recommendation task, the point- and pair-wise objectives are thus given by:

$$\mathcal{L}_{poi} = \sum\nolimits_{(u,i)\in\widetilde{O}} f(r_{ui}, \hat{r}_{ui}) + \lambda\Omega(\boldsymbol{\Theta}); \quad \mathcal{L}_{pai} = \sum\nolimits_{(u,i,j)\in\widetilde{O}} f(r_{uij}, \hat{r}_{uij}) + \lambda\Omega(\boldsymbol{\Theta}) \tag{1}$$

where $\widetilde{O} = \{O^{+} \cup O^{-}\}$ is the augmented dataset with the unobserved user-item pairs $O^{-} = \{(u, j)|r_{uj} = 0\}$ in addition to the observed user-item set $O^{+} = \{(u, i)|r_{ui} = 1\}$; $f(\cdot)$ is the loss function; $r_{ui}, \hat{r}_{ui}$ are the observed and estimated feedback of user u on item i, respectively; (u, i, j) is the triple meaning that user u prefers positive item i to negative item j; $r_{uij} = r_{ui} - r_{uj}, \hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}$; $\Omega(\boldsymbol{\Theta})$ is the regularization term; $\boldsymbol{\Theta}$ is the set of model parameters to be learnt. W.r.t. the loss function $f(\cdot)$, point-wise objective usually adopts square loss and cross-entropy (CE) loss, whereas pair-wise objective generally employs log loss and hinge loss:

$$\mathcal{L}_{poi} = \begin{cases} \text{Square} & f_{sl}(r_{ui}, \hat{r}_{ui}) = \frac{1}{2}(r_{ui} - \hat{r}_{ui})^2 \\ \text{CE} & f_{cl}(r_{ui}, \hat{r}_{ui}) = -r_{ui} log(\hat{r}_{ui}) - (1 - r_{ui}) log(1 - \hat{r}_{ui}) \end{cases}; \quad \mathcal{L}_{pai} = \begin{cases} \text{Log} & f_{ll}(r_{uij}, \hat{r}_{uij}) = log(1 + exp(-r_{uij} \cdot \hat{r}_{uij})) \\ \text{Hinge} & f_{hl}(r_{uij}, \hat{r}_{uij}) = max(0, 1 - r_{uij} \cdot \hat{r}_{uij}) \end{cases}$$

We follow majority studies [14, 32, 36, 37] and treat the unobserved feedback as negative feedback. We note that there may be different explanations behind the unobserved feedback [46], which we leave for further exploration. Table 2(a) shows the original objective functions used by BPRMF, BPRFM, NeuMF and SLIM (see formulas in the Additional Material of our GitHub repository). Besides, we vary different objective functions on these baselines to further investigate their respective impacts. Note that MostPop, ItemKNN and PureSVD do not have objective functions; we did not consider the square loss, as it is more suitable for rating prediction task instead of ranking problem [26]; and

(a) Objective functions of different baselines

Method	BPRMF	BPRFM	SLIM	NeuMF
Origin	$\mathcal{L}_{pai} + f_{ll}$	$\mathcal{L}_{pai} + f_{ll}$	$\mathcal{L}_{poi} + f_{sl}$	$\mathcal{L}_{poi} + f_{cl}$
To explore	$\mathcal{L}_{pai} + f_{hl}$ $\mathcal{L}_{poi} + f_{cl}$	$\mathcal{L}_{pai} + f_{hl}$ $\mathcal{L}_{poi} + f_{cl}$	– –	$\mathcal{L}_{pai} + f_{ll}$ $\mathcal{L}_{pai} + f_{hl}$

(b) Average size of training & test sets for each user.

Setting	ML-1M	Lastfm	Yelp	Epinions	Book-X	AMZe
origin	86, 64	39, 10	4, 2	35, 30	10, 5	2, 2
5-filter	86, 64	30, 8	13, 5	23, 13	23, 7	7, 3
10-filter	86, 64	27, 7	21, 8	20, 10	28, 8	12, 5

Table 2. For each record in Table (b), the 1st and 2nd numbers separated by comma denote the average sizes of training and test sets for each user, respectively.

we did not study the impacts of different objectives on SLIM due to its high complexity and low scalability, which will be discussed in Section 3.6.

2.2.5 Negative Sampling. As pointed out in Section 2.2.4, properly exploiting the unobserved feedback (treated as negative samples [14, 37]) helps learn users' relative preferences and benefits more accurate top-N recommendation. This can be further supported by the fact that 70% of the collected papers consider the unobserved feedback when designing objective functions regardless of point-wise and pair-wise ones. However, it is not practical to leverage all unobserved feedback in large volume, as most users only provide feedback for a small number of items. Negative sampling is, therefore, adopted to balance the efficiency and effectiveness.

Typically, there are different kinds of negative sampling strategies. For instance, **uniform sampler** [14], where all unobserved items of each user are sampled with an equal probability, has been adopted by almost all the papers in the collection on our observation. To better study the impact of negative sampling, we additionally consider and compare item popularity-based samplers, which have also been considered in recommendation [15, 41]: (1) **low-popularity sampler**: for each user, her unobserved items with a lower popularity are sampled with a higher probability. This is based upon the assumption that a user is less likely to prefer less popular items; and (2) **high-popularity sampler**: it is opposite to the low-popularity sampler, and the unobserved items of each user with a higher popularity are more probably to be sampled. The rationale behind is that: if a user provides no feedback for a quite popular item favored by a large number of users, it indicates that she may be really not into this item.

2.2.6 Data Splitting Methods. There are mainly two types of data splitting methods in the collected papers: **split-by-ratio** (61% of the papers) and **leave-one-out** (28% of the papers). In particular, split-by-ratio means that a proportion ρ (e.g., $\rho = 80\%$) of the dataset (i.e., user-item interaction records) is treated as training set, and the rest ($1 - \rho = 20\%$) as test set. While, leave-one-out refers to that for each user, only one record is kept for test and the remaining are for training. Besides, 5% of papers directly split training and test sets by a fixed timestamp, that is, the data before the fixed timestamp is used as training set, and the rest as test set; 6% of papers do not report their data splitting methods.

Although 61% of papers adopt split-by-ratio, they are quite different from each other due to: (1) different proportion settings, e.g., $\rho = 50\%, 60\%, 70\%, 80\%, 90\%$; (2) global- or user-level split. That is, some globally split the entire records into training and test sets regardless of different users; whilst others split training and test sets on the user basis; and (3) random- or time-aware split. Among papers exploiting split-by-ratio, 88% of papers merely randomly split the data, whereas 12% of papers split the data based on the timestamp, i.e., the earlier (e.g., $\rho = 80\%$) records as training and the later ones as test. In terms of leave-one-out, the split is generally on the user basis, and we find that timestamp is taken into account by 54% of papers with leave-one-out. Besides, to improve the test efficiency, they usually randomly sample a number of negative items (e.g., $neg\_test = 99, 100, 999, 1,000$) that are not interacted by each user, and then rank each test item among the ($neg\_test + 1$) items [14]. In our study, we follow the majority practice and select both random- and time-aware **split-by-ratio** at **global-level** with $\rho = 80\%$ as our data splitting method, and leave the exploration on leave-one-out as future work. Meanwhile, to speed up the test process, we randomly sample negative items for each

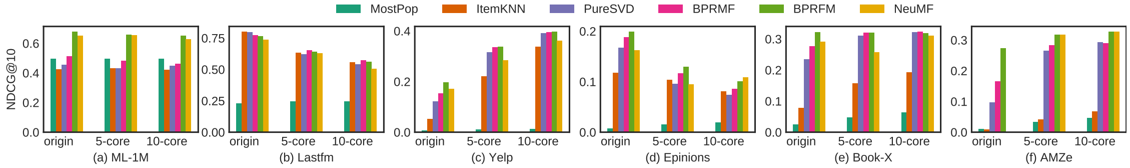


Fig. 2. Performance of different baselines w.r.t. time-aware split-by-ratio on the six datasets across origin, 5- and 10-filter settings.

user to ensure her test candidates to be 1, 000, and rank all test items among the 1, 000 items. Table 2(b) depicts the average number of test items for each user on the six datasets across origin, 5- and 10-filter settings, where all values are smaller than 100, indicating that 1, 000 test candidates is sufficient to examine the performance of recommenders.

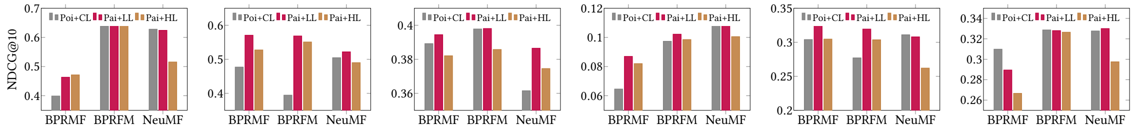
2.2.7 Evaluation Metrics. The evaluation metrics change a lot in different papers in the collection. Figure 1(d) depicts the popularity of the used evaluation metrics. We thus adopt the top-6 metrics covering 94% of the collected papers, meaning 94% of these papers adopt at least one of the six metrics. They are **Precision**, **Recall**, Mean Average Precision (**MAP**), Hit Ratio (**HR**), Mean Reciprocal Rank (**MRR**) and Normalized Discounted Cumulative Gain (**NDCG**), where the first four metrics measure whether a test item is present in the top-N recommendation list, whilst the latter two metrics accounts for the ranking positions of test items (see formulas in the Additional Material of our GitHub repository).

2.2.8 Hyper-parameter Tuning. Hyper-parameter tuning, including parameter validation and searching strategies, plays a vital role in the training process of recommendation approaches, and greatly influences the final performance.

Validation Strategy. Through the paper analysis, we notice that more than 37% of papers directly tune hyper-parameters according to the performance on the test set. That is to say, they use the same data to tune model parameters and evaluate model performance. Information may thus leak into the model and overfit the historical data. As a matter of fact, besides the split for training and test sets in cross-validation, an extra validation set should be retained to help tune the hyper-parameters, which is called **nested validation**[1]. With nested validation, the optimal hyper-parameter settings are obtained when the model achieves the best performance on the validation set. By doing so, the information leak issue is well avoided in the model training and evaluation process. In our study, we adopt the nested validation strategy, that is, in each fold of cross-validation, we further select from the training set **10%** of records as the validation set to tune hyper-parameters. Once the optimal settings for hyper-parameters are decided, we feed the whole training set (including the validation set) to train the final model and then report the performance on the test set. Due to the computational requirements of certain of baselines, we were unable to search in a reasonable amount of time the hyper-parameter space for a inner-loop of cross-validation for splitting the validation set from the training set.

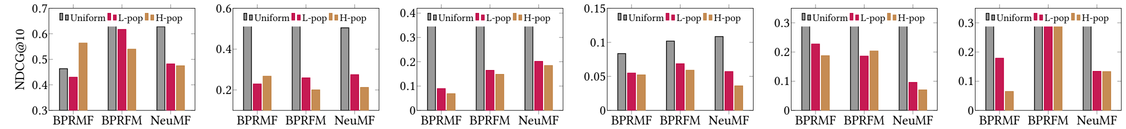
Searching Strategy. From our observation, almost all of the collected papers employ **grid search** [14, 37] to find out the optimal parameter settings. In particular, each hyper-parameter is provided with a set of possible values (i.e., search space) based on the prior-knowledge, and the optimal setting is then obtained by traversing the entire search space. Suppose a model has m parameters, where each parameter has an average of n possible values, the model needs to be executed for n^m times to find out optimal settings for all parameters. Therefore, grid search is more suitable for models with less parameters; otherwise, it may suffer from the combination explosion issue. To improve the parameter tuning efficiency, other strategies have been introduced. Given the search space of each parameter, **random search** [4] randomly chooses trials for a pre-defined times (e.g., 30) instead of traversing the entire search space. It is able to find models that are as good or slightly worse but within a smaller fraction of the computation time. On the contrary,

[1] https://scikit-learn.org/stable/auto_examples/model_selection/plot_nested_cross_validation_iris.html



(a) ML-1M (b) Lastfm (c) Yelp (d) Epinions (e) Book-X (f) AMZe

Fig. 3. Impacts of different objective functions on different baselines w.r.t. time-aware split-by-ratio across the six datasets.



(a) ML-1M (b) Lastfm (c) Yelp (d) Epinions (e) Book-X (f) AMZe

Fig. 4. Impacts of different negative samplers on different baselines w.r.t. time-aware split-by-ratio across the six datasets.

Bayesian HyperOpt [30] is not a brute force but more intelligent technique compared to grid and random search. It makes use of information from past trials to inform the next set of hyperparameters to explore, while not compromising the quality of the results [7]. To achieve an efficient hyper-parameter tuning, we adopt Bayesian HyperOpt to perform hyper-parameter optimization on **NDCG**, which is the most popular metrics among all evaluation metrics as shown in Figure 1(d). In this case, other metrics are expected to be simultaneously optimized with the optimal results on NDCG.

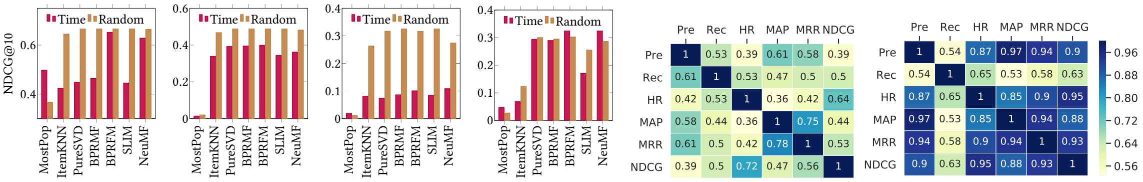
3 IMPACTS OF DIFFERENT FACTORS

3.1 Data Pre-processing

To study the impacts of different pre-processing strategies (origin, 5- and 10-filter), we adopt Bayesian HyperOpt to perform hyper-parameter optimization w.r.t. NDCG@10 for each baseline under each strategy on each dataset for 30 trails [7]. We keep the original objective functions for each baseline as shown in Table 2(a), employ the uniform sampler, and adopt time-aware split-by-ratio at global level ($\rho = 80\%$) as the data splitting method. Besides, 10% of the latest training set is held out as the validation set to tune the hyper-parameters. Once the optimal settings for hyper-parameters are decided, we feed the whole training set to train the final model and report the performance on the test set (without further statement, the subsequent studies in this section follow these settings). Figure 2 depicts the final results, where SLIM is omitted due to its extremely high computational complexity, and it is unable to complete in a reasonable amount of time (see Section 3.6). Meanwhile, the results of NeuMF on AMZe with origin setting are not available due to lack of computational memory. Owing to the space limitation, we only report the results on NDCG@10.

Overall, three different trends can be observed from the results: (1) the performance of different baselines keeps relatively stable on ML-1M with varied settings; (2) on Yelp, Book-X and AMZe, the performance of all baselines generally climbs up; and (3) a clear performance drop is observed on Lastfm and Epinions. The most probable explanation is that although the density of the datasets increases (origin ⟶ 5-filter ⟶ 10-filter) as shown in Table 1, the average length of the training sets for each user keeps stable on ML-1M (86); increases on Yelp, Book-X and AMZe; and decreases on Lastfm (39 ⟶ 30 ⟶ 27) and Epinions (35 ⟶ 23 ⟶ 20), as depicted by Table 2(b). The more training data per user, the better a model can be trained, meaning that the more accurate performance can be achieved, and *vice versa*.

Regarding to the performance comparison of different baselines, we notice that (1) in most cases, the MMs – MostPop performs the worst among all baselines, suggesting the importance of personalization in recommendation; and ItemKNN is defeated by the LFMs and DLMs, indicating the superiority of LFMs and DLMs on effective recommendation.



(a) ML-1M (b) Yelp (c) Epinions (d) AMZe (e) Correlation (f) Correlation

	Pre	Rec	HR	MAP	MRR	NDCG
Pre	1	0.53	0.39	0.61	0.58	0.39
Rec	0.61	1	0.53	0.47	0.5	0.5
HR	0.42	0.53	1	0.36	0.42	0.64
MAP	0.58	0.44	0.36	1	0.75	0.44
MRR	0.61	0.5	0.42	0.78	1	0.53
NDCG	0.39	0.5	0.72	0.47	0.56	1

	Pre	Rec	HR	MAP	MRR	NDCG
Pre	1	0.54	0.87	0.97	0.94	0.9
Rec	0.54	1	0.65	0.53	0.58	0.63
HR	0.87	0.65	1	0.85	0.9	0.95
MAP	0.97	0.53	0.85	1	0.94	0.88
MRR	0.94	0.58	0.9	0.94	1	0.93
NDCG	0.9	0.63	0.95	0.88	0.93	1

Fig. 5. (a-d) depict the performance of random- and time-aware split-by-ratio on different baselines across the four datasets; (e-f) show the correlations of evaluation metrics w.r.t. time-aware split-by-ratio, where ‘Pre’, ‘Rec’ denote Precision and Recall, respectively.

However, on ML-1M, the performance of MostPop exceeds that of ItemKNN, PureSVD and even BPRMF, demonstrating the potential of popularity in effective recommendation; and on Lastfm, ItemKNN achieves comparable even better performance in comparison with LFMs and DLMs. This implies that, the neighborhood-based idea, though simple, could be absorbed by LFMs and DLMs to further improve the recommendation accuracy [18]; (2) in terms of the three LFMs, BPRMF generally performs better than PureSVD but worse than BPRFM. Although PureSVD is simple – directly applying conventional sigular value decomposition on the user-item implicit interaction matrix, it sometimes can achieve comparable and even better performance when compared with BPRMF and BPRFM (see Lasfm-origin, Yelp-10-filter, Book-X-10-filter); (3) NeuMF, as the sole DLM among all baselines, performs comparably to BPRFM, and better than BPRMF on ML-1M and AMZe. However, on the rest four datasets, it generally underperforms BPRFM, BPRMF, and even PureSVD. This is consistent with the previous findings [7] that DLMs are not always better than traditional methods with well-tuned parameters, but mostly cost much more in training as verified by Table 3.

3.2 Objective Function

To examine the impacts of different objective functions, we adopt the optimal parameters for the baselines found on 10-filter setting in Section 3.1, and only vary objective functions for BPRMF, BPRFM and NeuMF (without further statement, the subsequent studies are based on 10-filter setting and adopt the corresponding optimal parameters found in Section 3.1). The results are depicted by Figure 3, where Poi+CL (point-wise cross entropy loss), Pai+LL (pair-wise log loss), Pai+HL (pair-wise hinge loss) correspond to $\mathcal{L}_{poi} + f_{cl}$, $\mathcal{L}_{pai} + f_{ll}$ and $\mathcal{L}_{pai} + f_{hl}$ in Table 2(a), respectively. Several conclusions can be drawn: (1) as a whole, Pai+LL generally achieves the best performance on the six datasets, but it is hard to compare the performance of Poi+CL and Pai+HL. For example, Poi+CL outperforms Pai+HL on AMZe, whereas cases are different on the other datasets; (2) from the perspective of different baselines, BPRMF usually achieves the best performance with Pai+LL; BPRFM is relatively less sensitive to different objectives, indicating its robustness; and NeuMF performs comparably with Poi+CL and Pai+LL, whilst obtains worse results with Pai+HL.

3.3 Negative Sampling

This subsection explores the impact of different negative samplers, i.e., uniform, low-popularity (L-pop) and high-popularity (H-pop) on BPRMF, BPRFM and NeuMF across the six datasets with 10-filter settings. To this end, we only vary negative samplers for the baselines while keeping other parameters fixed. To sum up, uniform sampler, though simple, achieves the best performance, as illustrated by Figure 4. In particular, a counter-intuitive observation is observed – the baselines with uniform sampler performs better than those with Low-pop. Intuitively, users may not tend to buy the less popular items, that is, the items with low popularity are more likely to be the negative items for users. However, it is overturned by the empirical results. Besides, L-pop slightly outperforms H-pop, which indicates that generally the popular items have a lower probability to become negative items than the less popular ones. Meanwhile, we also notice that on ML-1M the performance of H-pop for BPRMF far exceeds that of Uniform and L-pop, suggesting that a proper combination of uniform and popularity based samplers may potentially enhance the recommendation accuracy [16].

3.4 Data Splitting Methods

Here, we aim to test the impacts of different data splitting methods on the recommendation performance. For a practical study, we only compare random- and time-aware split-by-ratio at global level with $\rho = 80\%$, and leave other data splitting methods (e.g., leave-one-out) as future work. Figure 5(a-d) display the results of seven baselines on the four datasets with 10-filter setting, where we can clearly observe that baselines with random-aware split-by-ratio outperform those with time-aware split-by-ratio, especially on Epinions. The reason behind is that compared with random-aware split, time-aware split poses a stronger constraint on the pattern of training and test data, thus increasing the training difficulty. However, this is more close to the real recommendation scenario, which strives to infer future by history. Our study also implies that the empirical results disclosed in previous studies using random-aware split-by-ratio might be overestimated compared to those for real-world scenarios.

3.5 Evaluation Metrics

As discussed in Section 3.1, we adopt Bayesian HyperOpt to perform hyper-parameter optimization for 30 trials via optimizing NDCG. However, six metrics are utilized in this study, including Precision, Recall, HR, MAP, MRR and NDCG. The best hyper-parameter settings for optimal NDCG does not necessarily guarantee optimums w.r.t. the other five metrics. Hence, we study the correlation of different metrics when their respective optimums are achieved. In particular, for each baseline on each dataset with 10-filter setting, the Bayesian HyperOpt executes 30 trails, we thus have 30 entries for the validation performance of the baseline correspondingly, where each entry includes the results on the six metrics, e.g., [Precision: 0.24; Recall: 0.07; HR: 0.57; MAP: 0.17; MRR: 0.76; NDCG: 0.42]. Due to the optimal results for the six metrics may not achieve simultaneously, we select the optimal one among the 30 entries for each metric, and ultimately obtain six entries, where each entry records the best result on the corresponding metric.

Based on this, we pair-wisely calculate and record the times that any two of them (e.g., NDCG and HR) can achieve their best results simultaneously entry by entry. For example, given the optimal entry for NDCG, we will check whether the rest five metrics (e.g., HR) in this entry are optimal or not. If yes, we will add one at the corresponding position (NDCG, HR) of the correlation matrix; otherwise 0. The same rule is applied to the optimal entries for the other five metrics. Except MostPop, as it does not have any hyper-parameters, we accumulate the results of six baselines across the six datasets (6*6=36), and ultimately obtain their correlation matrix as illustrated by Figure 5(e), where all values are normalized into the range of $[0, 1]$ (divided by 36), and a darker color indicates a stronger correlation, that is, a higher probability of two metrics achieving their best results in the meanwhile. The results help verify our argument that best hyper-parameter settings for optimal NDCG cannot ensure optimal results for all the other five metrics. Moreover, we notice that the correlation matrix is asymmetrical. For instance, the correlation for (NDCG, HR, 0.72) is higher than (HR, NDCG, 0.64). That is to say, the probability of a model with best NDCG to achieve the best HR is higher than that of a model with best HR to reap the optimal NDCG. Meanwhile, the best MRR and MAP are more easily to be guaranteed concurrently with (MRR, MAP, 0.78) and (MAP, MRR, 0.75). Additionally, we examine the Kendall's correlation [35] among metrics in terms of indicating recommendation performance on the seven baselines across six datasets. The results are depicted by Figure 5(f), where a darker color (a stronger correlation) implies that the metrics produce more identical ranking. We see that Recall is noticeably poorly correlated with other metrics, whilst the ranking produced by

Table 3. Baseline comparisons on training time (seconds).

	MMs		**LFMs**				**DLMs**		**MMs**		**LFMs**				**DLMs**
	MostPop	ItemKNN	PureSVD	BPRMF	BPRFM	SLIM	NeuMF		MostPop	ItemKNN	PureSVD	BPRMF	BPRFM	SLIM	NeuMF
ML-1M	0.0149	50.976	0.7994	453.24	1,377.7	65.451	10,991	Epinions	0.0142	40.486	1.6707	1,696.8	2,081.1	827.29	8,476.5
Lastfm	0.0036	4.6749	0.0968	260.98	326.43	5.4544	397.08	Book-X	0.0151	43.137	1.4517	1,876.8	2,186.9	695.57	24,783
Yelp	0.0624	311.29	12.325	15,407	17,746	18,746	29,376	AMZe	0.0343	134.00	1.5266	2,807.7	4,674.6	7,410.0	47,103

the rest of the metrics shows a fairly strong correlation. In sum, a convincing and solid evaluation should be performed w.r.t. more diverse metrics.

3.6 Complexity Analysis

Table 3 shows the training time (with optimal hyper-parameters found by Bayesian HyperOpt under 10-filter setting) for the seven baselines on the six datasets. All the experiments are executed on an Amazon EC2 P2 instance (p2.8xlarge) with eight NVIDIA Tesla K80 Accelerators, each running a pair of NVIDIA GK210 GPUs and providing 12 Gib of memory; 16 CPU core (2.3 GHZ) sharing 488 GiB memory. Several major findings are noted: **(1)** MostPop is the fastest one in training, as it merely ranks all the items by the calculated popularity; **(2)** PureSVD is the runner-up with time complexity $O(\min\{m^2 f, n^2 f\})$, where m, n, f are the number of users, items and singular values, respectively. Compared with other LFMs and DLMs, it achieves a better balance between time complexity and accuracy. Particularly, it performs comparably and sometimes even better than BPRMF as depicted by Figure 2, while its training time is thousands times less than that of BPRMF; **(3)** although ItemKNN ranks third among all baselines with 10-filter setting, the time cost quadratically increases with origin setting due to its time complexity $O(mn^2)$. Besides, the similarity matrix also takes up huge memory, for example, on the original AMZe ($n \approx 10^6$), it will cost $(64\text{ bit} * 10^6 * 10^6)/10^{12} = 64T$ to save the similarity matrix. To ease this issue, we only keep the top-100 similar items for each target item in the memory; **(4)** the training time of BPRFM is slightly higher than that of BPRMF, as it requires updating global, user and item biases in addition to the user and item latent factors. The time complexity for both methods is $O(|\mathcal{R}|d)$, where $\mathcal{R}$ is the total number of observed feedback and d is the dimension of latent factors; **(5)** similar to ItemKNN, the time cost of SLIM with 10-filter setting is acceptable, while it tremendously increases with origin setting due to its time complexity $O(|R|n)$. Meanwhile, it also suffers from the huge memory cost issue because of the learned item similarity matrix. Hence, both ItemKNN and SLIM are not scalable for large-scale datasets; and **(6)** NeuMF, although yields comparable performance, costs much more training time than BPRFM, especially on much larger datasets. For example, on AMZe, the training time of NeuMF is ten times larger than that of BPRFM.

4 BENCHMARKING EVALUATION

4.1 Introduction to DaisyRec

To support our study, we build a user-friendly Python toolkit named as **DaisyRec**, where Daisy is short for 'Multi-**D**imension f**AI**rly comp**A**r**I**son for recommender **SY**stem'. Different from existing open-source libraries (e.g., LibRec [11], OpenRec [42] and DeepRec [44]), which mainly aim to reproduce various state-of-the-art recommendation algorithms, DaisyRec is designed with the goal of performing rigorous evaluation in recommendation. It is built upon the widely-used deep learning framework Pytorch (pytorch.org/), and Figure 6 depicts its overall structure consisting of three modules: Loader, Recommender and Evaluator. In particular, Loader mainly aims to: (1) load and pre-process the dataset; (2) split it into training and test sets based on the selected Splitter; (3) divide validation set from training set by choosing proper Splitter according to Step 2; (4) sample negative items for training by choosing samplers; and (5) convert the data into the specific format to fit the Recommender. Two components are included in Recommender, where 'Algorithm'

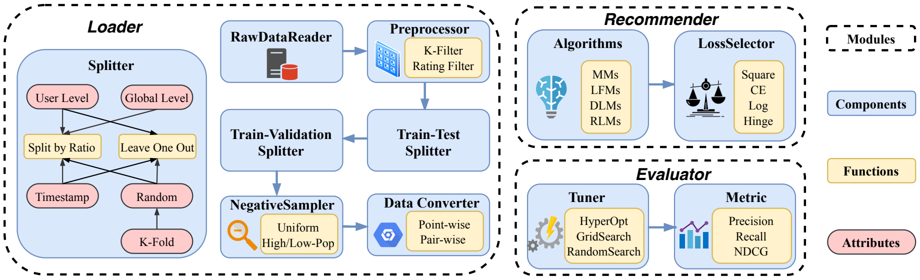


Fig. 6. Overall Structure of DaisyRec.

implements a number of state-of-the-arts, including MMs (MostPop, ItemKNN and UserKNN), LFMs (MF, BaisedMF, SVD++, RSVD2, PureSVD, SLIM, WRMF and FM), and RLMs (Item2Vec, Auto-Encoder, MLP, NeuMF, NFM and DeepFM); LossSelector makes it flexible to change different objective functions for the algorithms. Evaluator is equipped with Tuner and Metric, where the former helps accomplish hyper-parameter optimization and the latter implements the classic ranking metrics, e.g., Precision. To sum, all modules in DaisyRec are wrapped friendly for users to deploy, and new algorithms can be easily added into this extensible and adaptable framework.

Note that we are keeping DaisyRec updated and maintained. An updated version of DaisyRec will be relased soon, where several new influential factors are added, for example, (1) different parameter initialization methods; (2) different regularization terms to avoid over-fitting, e.g., L_1 and L_2 norms; (3) different rating prediction functions, e.g., consine simialrity, negative Euclidean distance and $L1$ distance; (4) more objective functions, e.g., top-1 loss [16]; and (5) more advanced baselines, e.g., NeuFM [13], CDAE [40], VAE [19], and so forth.

4.2 Standardized Procedures

As summarized in Section 2, there are a number of essential factors in the process of recommendation evaluation, which have been empirically analyzed in Section 3. To achieve a rigorous evaluation, we propose a series of standardized procedures and correspondingly call for endeavors of all researchers, aiming for effectively enhancing the standardization of recommendation evaluation. **(1)** It is impossible to evaluate either baselines or newly proposed methods on all public datasets covering each domain. However, at least one widely-used dataset discussed in Section 2 should be considered, especially for the papers evaluated on the private datasets (e.g., confidential data from commercial companies). Otherwise, the results could not be easily reproduced and compared by subsequent studies. **(2)** Section 3.1 verifies that different data pre-processing strategies impact the recommendation performance, as the scale of training and test data changes with different strategies. Besides original datasets, 5- and 10-filter settings are recommended to ease the data sparsity issue, and a clear description on data pre-processing details is indispensable. **(3)** The representative baselines with different types (MMs, LFMs and DLMs) in Section 2.2.3 are recommended to be selected and compared. As demonstrated in Section 3.1, the performance of different types of baselines vary a lot in different scenarios, that is, the MMs (e.g., MostPop and ItemKNN) and simple LFMs (e.g., PureSVD) sometimes even perform better than DLMs (e.g., NeuMF). The more diverse baselines are compared, the more comprehensive and reliable the evaluation is. **(4)** The recommendation performance is influenced by different objective functions as empirically proved by Section 3.2. For a fair comparison, it is better to evaluate all methods with different types of objective functions for a comprehensive evaluation, and thus better positioning a proposed method's contributions. **(5)** All the compared methods should adopt the same negative sampler, except the papers with the goal of proposing or studying different negative sampling strategies. **(6)** In terms

Table 4. Performance of seven baselines across six metrics on six datasets, where the best performance is underlined.

10-filter	ML-1M						Lastfm						Yelp					
	Pre	Rec	HR	MAP	MRR	NDCG	Pre	Rec	HR	MAP	MRR	NDCG	Pre	Rec	HR	MAP	MRR	NDCG
MostPop	0.2569	0.0532	0.7024	0.1745	0.8339	0.4975	0.0677	0.0880	0.4097	0.0349	0.2712	0.2476	0.0031	0.0040	0.0262	0.0012	0.0111	0.0137
ItemKNN	0.2208	0.0619	0.6381	0.1443	0.6821	0.4233	0.1851	0.2621	0.7840	0.1188	0.7708	0.5577	0.1174	0.1765	0.4882	0.0792	0.4600	0.3380
PureSVD	0.2270	0.0676	0.6900	0.1416	0.6984	0.4479	0.1862	0.2522	0.7556	0.1201	0.7673	0.5418	0.1508	0.2239	0.5837	0.0975	0.5493	0.3918
BPRMF	0.2711	0.0671	0.6454	0.1957	0.8453	0.4632	0.1987	0.2824	0.8167	0.1256	0.8058	0.5734	0.1538	0.2320	0.6034	0.0953	0.5439	0.3944
BPRFM	0.4082	0.1044	0.8534	0.3072	1.2834	0.6515	0.1974	0.2724	0.7976	0.1252	0.7930	0.5623	0.1579	0.2352	0.6154	0.0970	0.5483	0.3980
SLIM	0.2182	0.0642	0.6832	0.1377	0.6879	0.4450	0.2120	0.3064	0.8369	0.1380	0.8713	0.6018	0.1128	0.1655	0.4819	0.0778	0.4625	0.3432
NeuMF	0.3868	0.0915	0.8309	0.2912	1.2210	0.6278	0.1690	0.2382	0.7529	0.1001	0.6621	0.5044	0.1364	0.2090	0.5753	0.0794	0.4700	0.3615

10-filter	Epinions						Book-X						AMZe					
	Pre	Rec	HR	MAP	MRR	NDCG	Pre	Rec	HR	MAP	MRR	NDCG	Pre	Rec	HR	MAP	MRR	NDCG
MostPop	0.0040	0.0058	0.0370	0.0015	0.0139	0.0189	0.0134	0.0192	0.1081	0.0062	0.0547	0.0630	0.0085	0.0179	0.0793	0.0040	0.0387	0.0470
ItemKNN	0.0261	0.0315	0.1388	0.0150	0.0916	0.0813	0.0516	0.0786	0.2780	0.0335	0.2314	0.1921	0.0136	0.0312	0.1136	0.0067	0.0603	0.0679
PureSVD	0.0238	0.0285	0.1332	0.0124	0.0782	0.0736	0.1079	0.1622	0.4740	0.0695	0.4223	0.3216	0.0758	0.1781	0.4530	0.0429	0.3248	0.2935
BPRMF	0.0270	0.0341	0.1463	0.0149	0.0957	0.0862	0.1088	0.1905	0.5192	0.0617	0.3928	0.3241	0.0747	0.1735	0.4500	0.0416	0.3178	0.2894
BPRFM	0.0303	0.0357	0.1696	0.0161	0.1086	0.1006	0.1079	0.1842	0.5172	0.0602	0.3823	0.3185	0.0868	0.1908	0.4926	0.0509	0.3778	0.3280
SLIM	0.0260	0.0327	0.1423	0.0147	0.0936	0.0840	0.0753	0.1014	0.3475	0.0510	0.3264	0.2504	0.0342	0.0802	0.2426	0.0213	0.1769	0.1702
NeuMF	0.0302	0.0375	0.1772	0.0164	0.1160	0.1084	0.1057	0.1684	0.4915	0.0610	0.3831	0.3109	0.0866	0.1901	0.4921	0.0507	0.3768	0.3277

of the data splitting methods, both time-aware split-by-ratio and time-aware leave-one-out are recommended. By considering timestamp, the real recommendation scenario will be better simulated. W.r.t. split-by-ratio, both global- and user-level work well and $\rho = 80\%$ is recommended for a more feasible and convenient comparison. **(7)** At least two of the six discussed metrics in Section 2.2.7 should be adopted, where one (e.g., Precision, Recall, HR and MAP) is used to measure whether a test item is present on the top-N recommendation list, and the other (e.g., MRR and NDCG) helps measure the ranking positions of the recommended items. **(8)** Regarding to the most essential part – hyper-parameter tuning, a nested validation is mandatory, that is, retaining partial (e.g., 10%) training data as validation set. Bayesian HyperOpt, as a more intelligent parameter searching strategy, is recommended, and the search space should be kept the same for the shared parameters of different baselines. The number of trails (we set 30 in this study by following [7]) may be increased for further performance improvements. Most importantly, the optimal parameter settings should be well reported for reproduction. **(9)** The clear code, datasets and detailed parameter settings for each publication should be available for reproduction [23]. The conference venues could make them as necessities, measure the quality, and even require a short code demonstration along with each accepted paper during the conference.

4.3 Performance of Baselines

With the goal of providing a better reference for fair comparison, Table 4 shows the performance of seven baselines across six metrics on the six datasets under the 10-filter setting with $N = 10$. Due to space limitation, other results (e.g., origin and 5-filter settings with $N = 1, 10, 5, 20, 30, 50$) are available on our GitHub repository. As mentioned in Section 3.1, all the optimal hyper-parameters are found by adopting Bayesian HyperOpt to optimize NDCG for 30 trials, and the detailed parameter settings are summarized in the Additional Material of our GitHub repository. Similar observations discussed in Section 3.1 can be noted. Specifically, BPRFM achieves the best performance on ML-1M and AMZe; SLIM performs the best on Lastfm; NeuMF is the winner on Epinions; the best performance for Yelp is obtained by BPRFM w.r.t. (Precision, Recall, HR, NDCG) and PureSVD w.r.t. (MAP, MRR); while on Book-X, BPRMF achieves the best performance w.r.t. (Precision, Recall, HR, NDCG), and still PureSVD helps reach the best MAP and MRR.

5 RELATED WORK

While long been recognized as a key feature of scientific discoveries, reproducibility has been increasingly characterized as a crisis recently [1, 21]. It is becoming a primary concern in computer and information science, evidenced by the recently developed ACM policy on Artifact Review and Badging[2] and emerging efforts including seminars [10],

[2] https://www.acm.org/publications/policies/artifact-review-badging; see also SIGIR's implementation of the policy [9].

workshops [5], and focused tracks at major conferences [12]. Specific to recommender systems research, the discussions have been concentrated on the fairness of comparison between newly proposed and baseline methods [7, 25]. In very recent work, Dacrema et al. [7] find that neural models can hardly outperform fine-tuned memory- and latent factor-based methods, a similar finding also discovered by Rendle et al. [25].

Despite the importance, improving reproducibility in recommender systems research is highly challenging due to the many influential evaluation factors for recommendation performance. Said et al. [27] find large differences in the effectiveness of recommendation methods across different implementation frameworks as well as across evaluation datasets and metrics. A companion toolkit RiVal [28] was released to allow for the control of data splitting and evaluation metrics. Beel et al. [3] find a similar phenomenon in news and research paper recommendation and identify influential factors such as user characteristics and time of recommendation. Valcarce et al. [35] specifically study the properties of evaluation metrics for item ranking, marking precision as the most robust and NDCG presenting the highest discriminative power. More recently, Rendle et al. [25] demonstrate the importance of hyperparameter search in baseline methods, e.g., matrix factorization, and stress the need for standardized benchmarks where methods should be extensively tuned for fair comparison.

Existing benchmarks are, however, either restricted to pre-neural methods [27], a single evaluation factor [35], or rating prediction [25] which has been discouraged as a way to formulate the recommendation problem [20]; besides, all existing benchmarks consider two or three datasets (including [7]), ignoring the richness of available datasets often chosen by newly published work. Aimed for a full treatment of evaluation issues, our work takes a bottom-up approach analyzing an extensive amount of literature to search for important evaluation factors (e.g., objective function and negative sampling missing in existing benchmarks). We present a benchmark supported by an empirical study at a bigger-than-ever scale with the hope of laying a strong foundation for future research.

6 CONCLUSION

Due to lack of effective benchmarks, unreproducible evaluation and unfair comparison have become two major concerns in recommendation. This paper, therefore, aims to benchmark recommendation for reproducible evaluation and fair comparison. To this end, 85 recommendation papers published in the three recent years (2017-2019) from eight top tier conferences have been systematically reviewed, whereby we summarize the essential factors related to evaluation, e.g., data splitting methods, evaluation metrics and hyper-parameter tuning strategies, etc. Through an extensive empirical study, the impacts of different factors on evaluation are then comprehensively analyzed. Accordingly, we create benchmarks for rigorous evaluation by proposing standardized procedures and providing the performance of seven well-tuned state-of-the-art algorithms on six widely-used datasets across six metrics as a reference for later study. Lastly, a user-friendly Python toolkit – DaisyRec has been released from the angle of achieving rigorous evaluation in recommendation. For the future work, we plan to deepen our investigation by, for example, studying more baselines from other venues, and diving into more diverse recommendation tasks (e.g., session-aware).

7 ACKNOWLEDGMENTS

We gratefully acknowledge the support of National Natural Science Foundation of China (Grant No. 71601104, 71601116 and 71771141). This work was partly conducted within the Delta-NTU Corporate Lab for Cyber-Physical Systems with funding support from Delta Electronics Inc.

REFERENCES

[1] Monya Baker. 2016. Reproducibility crisis? *Nature* 533, 26 (2016), 353–66.

[2] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 1–6.

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

[4] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research (JMLR)* 13, 1 (2012), 281–305.

[5] Ryan Clancy, Nicola Ferro, Claudia Hauff, Jimmy Lin, Tetsuya Sakai, and Ze Zhong Wu. 2019. Overview of the 2019 Open-Source IR Replicability Challenge (OSIRRC 2019). In *OSIRRC@ SIGIR*. 1–7.

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

[7] 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 Thirteenth ACM Conference on Recommender Systems (RecSys)*. 101–109.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 248–255.

[9] Nicola Ferro and Diane Kelly. 2018. SIGIR initiative to implement ACM artifact review and badging. In *ACM SIGIR Forum*, Vol. 52. ACM New York, NY, USA, 4–10.

[10] Juliana Freire, Norbert Fuhr, and Andreas Rauber. 2016. Reproducibility of Data-Oriented Experiments in e-Science (Dagstuhl Seminar 16041). *Dagstuhl Reports* 6, 1 (2016), 108–159.

[11] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. 2015. LibRec: A Java Library for Recommender Systems.. In *The Twenty-third International Conference on User Modeling, Adaptation and Personalization (UMAP) Workshops*, Vol. 4.

[12] Allan Hanbury et al. 2015. *Proc. 37th European Conference on IR Research*. Vol. 9022. Springer.

[13] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the Fortieth International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 355–364.

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

[15] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the Thirty-ninth International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 549–558.

[16] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the Twenty-seventh ACM International Conference on Information and Knowledge Management (CIKM)*. 843–852.

[17] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining (ICDM)*. IEEE, 263–272.

[18] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys)*. 306–310.

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

[20] Sean M McNee, John Riedl, and Joseph A Konstan. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*. 1097–1101.

[21] Marcus R Munafò, Brian A Nosek, Dorothy VM Bishop, Katherine S Button, Christopher D Chambers, Nathalie Percie Du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J Ware, and John PA Ioannidis. 2017. A manifesto for reproducible science. *Nature human behaviour* 1, 1 (2017), 1–9.

[22] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE Eleventh International Conference on Data Mining (ICDM)*. IEEE, 497–506.

[23] Edward Raff. 2019. A Step Toward Quantifying Independently Reproducible Machine Learning Research. In *Advances in Neural Information Processing Systems (NIPS)*. 5485–5495.

[24] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining (ICDM)*. IEEE, 995–1000.

[25] Steffen Rendle et al. 2019. On the difficulty of evaluating baselines: A study on recommender Systems. *arXiv preprint arXiv:1905.01395* (2019).

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Fourteenth International Conference on Intelligent User Interfaces (IUI)*. 452–461.

[27] Alan Said and Alejandro Bellogín. 2014. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the Eighth ACM Conference on Recommender Systems (RecSys)*. 129–136.

[28] Alan Said and Alejandro Bellogín. 2014. Rival: a toolkit to foster reproducibility in recommender system evaluation. In *Proceedings of the Eighth ACM Conference on Recommender Systems (RecSys)*. 371–372.

[29] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International Conference on World Wide Web (WWW)*. 285–295.

[30] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*. 2951–2959.

[31] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications (ECRA)* 37 (2019), 100879.

[32] 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 Twelfth ACM Conference on Recommender Systems (RecSys)*. 297–305.

[33] Jiliang Tang, Huiji Gao, Huan Liu, and Atish Das Sarma. 2012. eTrust: Understanding trust evolution in an online world. In *Proceedings of the Eighteenth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 253–261.

[34] Xiaoli Tang, Tengyun Wang, Haizhi Yang, and Hengjie Song. 2019. AKUPM: Attention-enhanced knowledge-aware user preference model for recommendation. In *Proceedings of the Twenty-fifth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1891–1899.

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

[36] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *Proceedings of the Twenty-eighth International Conference on World Wide Web (WWW)*. 2000–2010.

[37] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the Twenty-fifth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 950–958.

[38] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the Forty-second International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 165–174.

[39] Ga Wu, Maksims Volkovs, Chee Loong Soon, Scott Sanner, and Himanshu Rai. 2019. Noise contrastive estimation for one-class collaborative filtering. In *Proceedings of the Forty-second International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 135–144.

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

[41] Fengli Xu, Jianxun Lian, Zhenyu Han, Yong Li, Yujian Xu, and Xing Xie. 2019. Relation-aware graph convolutional networks for agent-initiated social e-commerce recommendation. In *Proceedings of the Twenty-eighth ACM International Conference on Information and Knowledge Management (CIKM)*. 529–538.

[42] Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh, and Deborah Estrin. 2018. Openrec: A modular framework for extensible and adaptable recommendation algorithms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*. 664–672.

[43] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the Twenty-second ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 353–362.

[44] Shuai Zhang, Yi Tay, Lina Yao, Bin Wu, and Aixin Sun. 2019. DeepRec: An Open-source Toolkit for Deep Learning based Recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, (IJCAI)*. ijcai.org, 6581–6583.

[45] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[46] Qian Zhao, Martijn C Willemsen, Gediminas Adomavicius, F Maxwell Harper, and Joseph A Konstan. 2018. Interpreting user inaction in recommender systems. In *Proceedings of the Twelfth ACM Conference on Recommender Systems (RecSys)*. 40–48.

[47] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the Fourteenth International Conference on World Wide Web (WWW)*. 22–32.