Towards a More User-Friendly and Easy-to-Use Benchmark Library for Recommender Systems

Lanling Xu[◇†]
GSAI, Renmin University of China
Beijing, China
xulanling@ruc.edu.cn

Zhen Tian[†]
GSAI, Renmin University of China
Beijing, China
chenyuwuxinn@gmail.com

Gaowei Zhang
GSAI, Renmin University of China
Beijing, China
zgw15630559577@163.com

Junjie Zhang
GSAI, Renmin University of China
Beijing, China
junjie.zhang@ruc.edu.cn

Lei Wang
GSAI, Renmin University of China
Beijing, China
zxcptss@gmail.com

Bowen Zheng
GSAI, Renmin University of China
Beijing, China
bwzheng0324@gmail.com

Yifan Li
GSAI, Renmin University of China
Beijing, China
liyifan0925@gmail.com

Jiakai Tang
GSAI, Renmin University of China
Beijing, China
tangjiakai5704@ruc.edu.cn

Zeyu Zhang
GSAI, Renmin University of China
Beijing, China
zeyuzhang@ruc.edu.cn

Yupeng Hou
GSAI, Renmin University of China
Beijing, China
houyupeng@ruc.edu.cn

Xingyu Pan
SI, Renmin University of China
Beijing, China
xy_pan@foxmail.com

Wayne Xin Zhao[✉]
GSAI, Renmin University of China
Beijing, China
batmanfly@gmail.com

Xu Chen[✉]
GSAI, Renmin University of China
Beijing, China
successcx@gmail.com

Ji-Rong Wen
GSAI, SI, Renmin University of China
Beijing, China
jrwen@ruc.edu.cn

ABSTRACT

In recent years, the reproducibility of recommendation models has become a severe concern in recommender systems. In light of this challenge, we have previously released a unified, comprehensive and efficient recommendation library called *RecBole*. With the increasing number of users, we have received a number of suggestions and update requests. This motivates us to make further improvements on our library, so as to meet the user requirements and contribute to the research community. In this paper, we present a significant update of RecBole, making it more user-friendly and easy-to-use as a comprehensive benchmark library for recommendation. More specifically, the highlights of this update are summarized as: (1) we include more benchmark models and datasets, improve the benchmark framework in terms of data processing, training and evaluation, and release reproducible configurations to benchmark the recommendation models; (2) we upgrade the user friendliness of our library by providing more detailed documentation and well-organized frequently asked questions, and (3) we propose several development guidelines for the open-source library developers. Our update is released at the link: https://github.com/RUCAIBox/RecBole#update.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender systems, Open-source library, Benchmark

ACM Reference Format:
Lanling Xu[◇†], Zhen Tian[†], Gaowei Zhang, Junjie Zhang, Lei Wang, Bowen Zheng, Yifan Li, Jiakai Tang, Zeyu Zhang, Yupeng Hou, Xingyu Pan, Wayne Xin Zhao[✉], Xu Chen[✉], and Ji-Rong Wen. 2023. Towards a More User-Friendly and Easy-to-Use Benchmark Library for Recommender Systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539618.3591889

◇ GSAI and SI are the abbreviations for Gaoling School of Artificial Intelligence and School of Information, respectively. All the authors are also with Beijing Key Laboratory of Big Data Management and Analysis Methods.
† Both authors contributed equally to this work.
✉ Corresponding authors.

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591889

Table 1: Comparison of different versions of RecBole in different aspects.

Aspect	**RecBole 1.0** [48]	**RecBole 2.0** [46]	**This update**
Recommendation tasks	4 categories	3 topics and 5 packages	4 categories
Models and datasets	73 models and 28 datasets	65 models and 8 new datasets	86 models and 41 datasets
Data structure	Implemented Dataset and Dataloader	Task-oriented	Compatible data module inherited from PyTorch
Continuous features	Field embedding	Field embedding	Field embedding and discretization
GPU-accelerated execution	Single-GPU utilization	Single-GPU utilization	Multi-GPU and mixed precision training
Hyper-parameter tuning	Serial gradient search	Serial gradient search	Three search methods in both serial and parallel
Significance test	-	-	Available interface
Benchmark results	-	Partially public (GNN and CDR)	Benchmark configurations on 82 models
Friendly usage	Documentation	Documentation	Improved documentation and FAQ page

1 INTRODUCTION

In the literature of recommender systems, the reproducibility of recommendation models has long been a severe concern [7]. For alleviating this issue, a number of open-source benchmark libraries [25, 28, 32, 48] have been developed for facilitating the research on recommender systems. Among these benchmark libraries, RecBole [46, 48] is featured with the unified benchmark framework, standardized models and datasets, extensive evaluation protocols, efficient training process and user-friendly documentation. Since the first release in 2020, RecBole has received over 2500 stars and 490 forks on GitHub, and our team is committed to solving the regular usage problems by handling more than 600 issues and 950 pull requests.

As a comprehensive recommendation library, RecBole not only keeps up with the most recent advances on recommender systems, but also continually optimizes its design for more flexible, easier use. Inspired by this motivation, this paper presents an important update of RecBole to make it more user-friendly and easy-to-use for academic research, featured by a *well-established benchmark library* for recommendation. Built on the previous versions of RecBole [46, 48], we make significant extensions in three major aspects, namely the models/datasets, the framework, and the configurations. First, based on previous efforts in RecBole, we include more benchmark models and datasets to meet the latest needs of users. Secondly, we improve the benchmark framework by including commonly used data processing methods and efficient training and evaluation APIs, and also provide more support for result analysis and use (*e.g.,* significance test and LaTeX-code generation). Thirdly, in order to improve the user experience, we provide more comprehensive project pages and documentation. With these resources, users can quickly reproduce the benchmark results and find the solutions to the frequently encountered issues at a glance. Besides the three aspects, we systematically investigate the user needs with recommendation libraries, and conduct detailed analysis with both GitHub issues and user questionnaires, which derive a series of useful guidance for developing recommendation libraries.

To summarize, the spotlights of this update are given in the following (see a detailed comparison in Table 1):

• **Updates to benchmark the recommendation models**. To facilitate researchers of recommender systems in a more easy-to-use way, we improve RecBole in three major aspects, *i.e.,* the models/-datasets, the framework, and the configurations. Firstly, for benchmark models and datasets, the number of models is increased from 73 to 86 (the main branch), and 13 new public datasets are added to our library on the basis of 28 existing processed ones, which can support the research on up-to-date advances in recommender systems. Secondly, for benchmark framework, our improvements can be featured in four core parts including flexible data processing, efficient training and evaluation, intelligent hyper-parameter tuning and improved performance comparison. To satisfy diverse needs for data processing, we reframe the overall data flow with PyTorch [23] towards a compatible data module. Meanwhile, considering the varied features of different recommendation tasks, we add more task-oriented data processing methods. To deal with large-scale recommendation datasets more efficiently, we significantly improve the GPU-based acceleration in this update. Specifically, we implement three new strategies, *i.e.,* multi-GPU training and evaluation, mixed precision training and intelligent hyper-parameter tuning. To provide a reliable performance comparison between models, we further develop two utility functions to implement significance test and generate LaTeX-format results, which improve the efficiency of academic research. Thirdly, for benchmark configurations, we provide the hyper-parameter selection range and recommended configurations for each model on three datasets respectively. With these benchmark configurations, researchers can easily reproduce and compare baseline models in our library.

• **Updates to improve the user friendliness**. To make our library more user-friendly, we update the website and documentation with detailed descriptions. In particular, we enrich our website by including new features in RecBole 2.0 [46] and visualization of benchmark configurations. Also, we update our tutorial by providing more practical examples of the customized training strategy, multi-GPU training cases and detailed running steps. With this updated documentation, new users can get started with our library quickly. Besides, we also develop a FAQ page based on the existing GitHub issues of RecBole. From this page, users of our library can browse the enhancement suggestions, use problems, academic discussions and experimental guidance at a time, which can avoid repeated issues and accelerate issue resolving.

• **Updates to summarize the development guidelines**. Recent years have witnessed many promising recommendation libraries with different specific advantages. In this paper, we further investigate what real users expect for an ideal recommendation library. In specific, we analyze the real user requirements from two aspects, *i.e.,* the historical issues and investigation questionnaires. These guidelines are useful to develop new libraries or improve existing ones for the open-source developers.

Table 2: Newly implemented models in RecBole on 4 tasks.

Tasks	Models	Conference	Year
General Recommendation	NCEPLRec [40]	SIGIR	2019
	ADMM SLIM [30]	WSDM	2020
	SGL [41]	SIGIR	2021
	SimpleX [17]	CIKM	2021
	NCL [16]	WWW	2022
Context-aware Recommendation	Fi-GNN [15]	CIKM	2019
	DCN-V2 [36]	WWW	2021
	KD_DAGFM [34]	WSDM	2023
Sequential Recommendation	SINE [31]	WSDM	2021
	LightSANs [6]	SIGIR	2021
	CORE [13]	SIGIR	2022
Knowledge-aware Recommendation	KGIN [39]	WWW	2021
	MCCLK [50]	SIGIR	2022

Table 3: Newly collected datasets in RecBole.

Datasets	#Users	#Items	#Interactions
Amazon (2018)	-	-	-
Alibaba-iFashion	3,569,112	4,463,302	191,394,393
AliEC	491,647	240,130	1,366,056
BeerAdvocate	33,388	66,055	1,586,614
Behance	63,497	178,788	1,000,000
DianPing	542,706	243,247	4,422,473
EndoMondo	1,104	253,020	253,020
Food	226,570	231,637	1,132,367
GoodReads	876,145	2,360,650	228,648,342
KGRec-music	5,199	8,640	751,531
KGRec-sound	20,000	21,533	2,117,698
ModCloth	47,958	1,378	82,790
RateBeer	29,265	110,369	2,924,163
RentTheRunway	105,571	5,850	192,544
Twitch	15,524,309	6,161,666	474,676,929
Yelp (4 versions)	5,556,436	539,254	28,908,240

2 BENCHMARK LIBRARY - RECBOLE

To standardize the implementation and evaluation for recommendation, a library generally needs to consider the following three key aspects, including (1) *benchmark models and datasets* that provide users with comprehensive baseline models and processed datasets, (2) *benchmark framework* that unifies the pre-processing, training and evaluation process of different models towards reliable performance comparison, and (3) *benchmark configurations* that enhance the reproducibility of model results with well-tuned hyper-parameters. Based on previous efforts of RecBole [46, 48], we have further upgraded our benchmark library in the latest version, which is featured in three aspects as introduced in the following sections.

2.1 Benchmark Models and Datasets

The rapid development of recommender systems has promoted more advanced recommendation models. Meanwhile, the demand for more comprehensive datasets is also increasing. In the latest version of RecBole, we have systematically expanded the models and datasets according to the recent requests of researchers. In specific, the number of models is increased from 73 to 86 (the main branch), and there are additional 13 datasets in our latest library.

2.1.1 Up-to-date Recommendation Models. Since the first release of RecBole in November 2020, we have been updating the implemented models considering representativeness, recency and usability. In RecBole, we divide the recommendation models into four categories, including general recommendation, sequential recommendation, context-aware recommendation and knowledge-aware recommendation [48]. In addition, we have implemented the social recommendation, cross-domain recommendation and fairness-aware recommendation in RecBole 2.0 [46]. So far, there are totally more than one hundred models included in RecBole [46, 48]. In this update, we further extend the four categories of models with rigorous evaluation on benchmark datasets. In terms of the selected models to be implemented, we have fully considered the feedback and suggestions of users from GitHub and email. Furthermore, we have carefully surveyed the latest literature related to recommender systems at the link: https://github.com/RUCAIBox/Awesome-RSPapers, in order to summarize the representative methods and techniques for recommendation. As presented in Table 2, we implement 13 up-to-date recommendation recommendation models, which involves a variety of newly popular techniques such as graph neural networks [13, 15, 39], contrastive learning [16, 41, 50] and knowledge distillation [34]. As a continuously updated open-source library, our team will continue to follow up the progress of recommender systems, and we also call for open-source contributors to work with us to implement more benchmark models in the future.

2.1.2 Extended Benchmark Datasets. In the field of recommender systems, experimental datasets vary in different papers with inconsistent pre-processing methods. Thus, the reported results of different papers may not be directly comparable. To overcome this issue, RecBole utilizes the atomic files to format the raw datasets in a unified way, which lays the foundation for a fair comparison. To ease the redundant work of researchers in collecting and processing datasets, we further add 13 new public datasets in the format of atmoic files on the basis of 28 existing processed ones. For details of our datasets and scripts, please refer to our dataset repository at the link: https://github.com/RUCAIBox/RecSysDatasets. As shown in Table 3, these new datasets have the following features:

• **Multi-domain datasets**. Recommender systems have a wide range of applications, and the newly added datasets for RecBole cover the categories of e-commerce, books, movies, clothes and restaurants, which enriches the research scenarios of our library and facilities relevant studies such as cross-domain recommendation.

• **Multi-version datasets**. For the same recommendation dataset or data on the same platform, there may be multiple versions with varied sizes and time periods. For example, the series of MovieLens datasets [10] include MovieLens-100K, MovieLens-1M, MovieLens-10M and MovieLens-20M, which are all processed in our repository. To further clarify multi-version datasets, we collect different versions of two commonly used datasets in the literature, *i.e.,* Yelp [33] datasets released in 2018, 2020, 2021 and 2022, and Amazon review

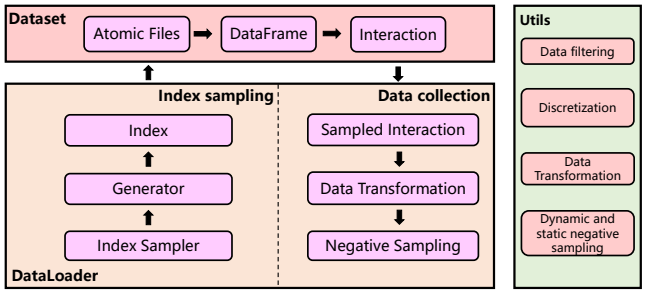

Figure 1: The overall framework of data module in RecBole.

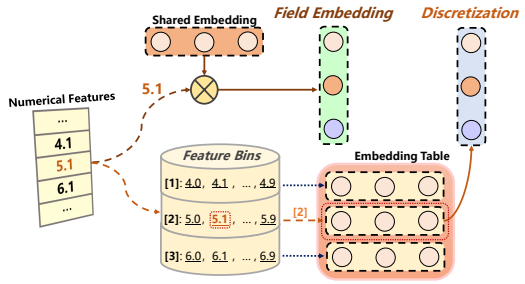

Figure 2: The unified embedding protocol for numerical fields.

datasets [19] of 24 sub-categories in 2014 and 29 sub-categories in 2018. With this improvement, researchers can easily select multiple versions of benchmark datasets from the classic to the latest one, which sheds light on the sustainability of benchmark datasets.

• **Rich-context datasets**. In addition to the interaction data, the context information of the users and items (*e.g.,* item attributes) is important for content or context-based recommendation tasks (*e.g.,* click through rate predictions). Considering this requirement, most of the updated datasets contain additional feature information for users or items, which can well support the research on context-based tasks (*e.g.,* fairness-aware recommendation).

2.2 Benchmark Framework Improvements

As a unified benchmark framework, the design of RecBole [46, 48] is decoupled in three core parts including extensible data structure, encapsulated training module and built-in evaluation protocols, which are convenient to use and customize. In this update, we further improve the benchmark framework in four major aspects, namely flexible data pre-processing, efficient training and evaluation, intelligent hyper-parameter tuning and better performance comparison. These four aspects are highly requested by the library users, which can largely improve the usability of our library.

2.2.1 Flexible Data Processing. Since recommender systems deal with diverse task scenarios of varied data formats, data processing has been a key step in facilitating the use of the benchmark library. In previous versions of RecBole [46, 48], we have designed an extensible data structure and provided a unified data processing pipeline. We further improve this pipeline by supporting more compatible data module and more task-oriented data processing as follows.

Compatible data module. To support multiple types of recommendation tasks in a unified way, we previously developed self-defined Dataset and DataLoader tailored to recommendation tasks, while customized data modules were not compatible with the built-in interfaces of PyTorch [23]. To better reuse the mainstream data modules supported by the mature, well-maintained PyTorch, we reconstruct a more general and compatible data module. As shown in Figure 1, our latest data module directly inherits torch.utils.data.Dataset and torch.utils.data.DataLoader to derive the two python classes respectively, *i.e.,* Dataset and DataLoader, as follows.

• The class Dataset has the following data flow: *Atomic Files* → *DataFrame* → *Interaction*, where the formatted dataset files are loaded and processed to derive the internal data structure Interaction. During the process from atomic files to pandas.DataFrame, we incorporate a series of data filtering and feature processing methods to obtain the target instances, which can be subsequently transformed to Interaction for unified iteration and indexing.

• The class DataLoader provides an iterable method upon the given class Dataset. For compatibility, we divide the process of each iteration into two stages: *index sampling* and *data collection*. The former generates the sampled index of Interaction, which is mainly implemented by the API provided by the base class of DataLoader in PyTorch. The latter is implemented in the interface collate_fn(·), which first retrieves the sampled Interaction based on the sampled index. To support various recommendation tasks, we apply a series of flexible data transformation and negative sampling strategies to generate the batch data for training and evaluation.

Discretization of numerical features. For numerical features, we support two mainstream representation approaches (*i.e., field embedding* and *discretization*) in the latest version of RecBole, which are widely used in existing studies [9]. For the former, *field embedding* is a commonly used approach in academia, where features in the same field share a uniform embedding and representations are obtained by multiplying their scale values. For the latter, recent industrial recommender systems tend to utilize *discretization* methods to handle numerical features. By discretization, continuous numerical features are transformed to different discrete value bins. Thus, we implement two widely used discretization methods, *i.e., equal distance discretization* and *logarithm discretization* as follows.

• *Equal distance discretization* divides the numerical features into multiple bins with equal length, which can be defined as:

$$\hat{x}_i = \text{floor}\left(\frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}} \cdot N_i\right), \tag{1}$$

where $[x_i^{max}, x_i^{min}]$ denotes the feasible range of the i-th numerical field, N_i denotes the number of discretization buckets of the i-th numerical field, and floor(·) is the rounding-down operation.

• *Logarithm discretization* is proposed by the champion of Criteo advertiser prediction in Kaggle [9]. By logarithm discretization, each numerical feature is transformed as:

$$\hat{x}_i = \text{floor}\big(\log(x_i)^2\big). \tag{2}$$

With this enhanced data module, our library can well support the representations of numerical features. As shown in Figure 2, each numerical feature can be represented in the above two ways:

```
from recbole.quick_start import run_recboles
import torch.multiprocessing as mp
args = dict(
    # General parameters in original RecBole
    model = 'BPR',
    dataset = 'ml-100k',
    config_file_list = ['test.yaml'],

    # Parameters for distributed training
    ip = 'IP address of the master node',
    port = 'Port number of the master node',
    world_size = 'Number of ranks in the global',
    nproc = 'Number of processes on current node',
    group_offset = 'Offset of the global rank'
)
mp.spawn(run_recboles, args=(args.model, args.dataset, \
    config_file_list, args.ip, args.port, args.world_size, \
    args.nproc, args.group_offset), nprocs=args.nproc)
```

Figure 3: Usage example of distributed training.

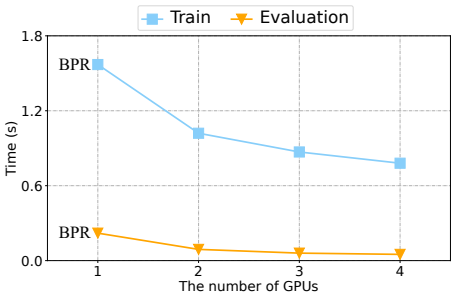

Figure 4: Efficiency comparison with different number of GPUs.

for the field embedding, we multiply the shared embedding with the feature value. While for discretization, we employ the assigned bin number to lookup the embedding table. Further, these two kinds of embeddings can be combined as the final feature representation.

Customized transformation for sequence data. Transformers are widely used in recommender systems, while most of existing works extend the original model with specific sequence transformations or augmentation techniques. Considering this, we add a new variable "transform" in the dataloader to perform useful transformations for sequential models, which are independent of model implementation. Specifically, we add four major augmentation operations for the processing procedure of data [42].

• MaskItemSequence is the token masking operation proposed in natural language processing, which has also been used in recommender systems such as S^3Rec [49]. Here, we introduce this transformation method for interaction sequences. For each user's historical sequence, this method replaces a proportion of items by a special symbol [mask] to generate a masked sequence.

• FlexiblePadLocation provides an option to specify the position of [pad] to align the sequence length, and users can choose to place it at the beginning or at the end of the augmented sequence.

• CropItemSequence [42] generates local views on historical interaction sequences by selecting a continuous sub-sequence.

• ReorderItemSequence alters the order of items in the user sequence by randomly shuffling their positions.

Besides, as a user-defined interface, UserDefinedTransform is reserved to facilitate further development by the users themselves.

Filtering and augmentation for knowledge graphs. As for the knowledge-aware models, since a knowledge graph is very large, we add a new mechanism to support the filtering process of entities and relations in the triples. Besides, as inverse relations are often used in knowledge-aware models [37, 39], we also provide a way for adding inverse relations. Considering the two requirements, we implement two specific methods for pre-processing knowledge graphs: (1) k-core filtering is performed on the nodes and relations to mitigate the noise problems, and (2) users can choose whether to construct the inverse relations for triples as data augmentation.

Dynamic and static negative sampling. Recent studies have shown that negative sampling [17] largely influences the model performance. While, existing recommendation libraries either ignore the sample selection strategies, or only provide static negative samplers, which are incapable to support recommendation models with specific negative sampling strategies. Considering the above issues, we add dynamic negative sampling method [43] in our implementation, and support three commonly used negative sampling methods, including random negative sampling (RNS) [26], popularity-biased negative sampling (PNS) [4] and dynamic negative sampling (DNS) [43]. In terms of PNS, we borrow the idea from word vector representation and set a hyper-parameter α to control the impact of popularity on negative sampling following [4]. Formally, the probability of an item to be sampled as a negative for a user is $p(i) \propto deg(i)^{\alpha}$, where $deg(i)$ is the total interaction number for item i by all the users. By setting different values for α, we can implement both low-popularity and high-popularity samplers.

2.2.2 Efficient Training and Evaluation. Recently, recommendation models have been largely enhancd by incorporating advanced techniques, such as graph neural networks [15, 41] and self-supervised learning [16, 41], which call for more efficient implementations for both training and evaluation. Meanwhile, side information such as knowledge graphs [39, 50] also increases the burden of data processing. To further improve the utilization of computing resources, we implement two practical and efficient methods, *i.e., distributed parallel* and *mixed precision training*.

Distributed parallel. Efficiency is an important factor to consider for a recommendation library. In this update, we support distributed computing by the multi-GPU and multi-machine method based on torch.nn.parallel.DistributedDataParallel, which can accelerate model training and evaluation. In addition to these general settings, users can customize the environment for distributed computing with minor modifications (*e.g.,* ip, port, world_size, nproc, and group_offset), and we present a usage example in Figure 3. To verify the effectiveness of our distributed parallel module, we empirically analyze the training and evaluation costs of the BPR model [26] on MovieLens-1M [10] dataset with different number of GPUs. As shown in Figure 4, as the number of GPUs increases, the training and evaluation time gradually decreases, showing the acceleration effect of distributed computing.

Mixed precision training. Mixed precision training is an optimization technique for model training that combines half-precision (FP16) and single-precision (FP32) for improving efficiency and reducing memory cost. Recently, GPUs using the NVIDIA Tensor

Table 4: Performance comparison of CFKG and NNCF *without* and *with* mixed precision training (MPT) on MovieLens-1M dataset, where TT denotes training time.

Model	MPT	Recall@10	NDCG@10	TT (sec/epoch)
CFKG	*w/o*	0.1708	0.2333	3.90
	w/	0.1708	0.2333	3.41
NNCF	*w/o*	0.1753	0.2446	10.30
	w/	0.1770	0.2471	5.47

Table 5: Performance comparison of hyper-tuning strategies for LightGCN and KGAT. ST denotes searching time, GS denotes Grid Search, RS denotes Random Search, and BH denotes Bayesian Hyper-opt.

Model	Strategy	ST (seconds)	Recall@10	NDCG@10
LightGCN	GS	21,875.00	0.1853	0.2487
	RS	15,611.00	0.1851	0.2498
	BH	16,073.00	0.1823	0.2476
KGAT	GS	198,942.00	0.1867	0.2521
	RS	142,813.00	0.1816	0.2456
	BH	150,900.00	0.1845	0.2498

```
python run_hyper.py --tool=Ray
python run_hyper.py --tool=Hyperopt
```

Figure 5: Usage examples of two hyper-tuning tools.

Cores [18] have shown that FP16 throughput is 2 to 8 times higher than that of FP32 [20]. Furthermore, the storage size required for FP16 is the half of that for FP32, significantly reducing the memory cost. To implement mixed precision training, we use torch.autocast during the loss optimization to automatically convert weights and gradients from FP32 to FP16. We also employ torch.cuda.GradScaler to scale the loss and avoid overflow, as the range of FP16 is smaller than that of FP32, which can lead to overflow if handled improperly. As shown in Table 4, we conduct experiments to verify the performance of CFKG [1] and NNCF [2] *with* and *without* mixed precision training on MovieLens-1M dataset. With mixed precision training, the training speed is increased by 14.4% on CFKG and 88.3% on NNCF, and the recommendation performance is retained.

2.2.3 Intelligent Hyper-Parameter Tuning. Neural recommendation models often rely on the hyper-parameter tuning to find the optimal configurations. Compared with the previous version of RecBole, which mainly supports *grid search*, this update provides users with more types of hyper-parameter tuning strategies, *i.e.,* *Random Search* [3] and *Bayesian Hyper-opt* [29]. For detailed analysis, we conduct experiments to compare the performance of three tuning strategies. Table 5 summarizes the performance of LightGCN [11] and KGAT [37] tuned with three strategies on the MovieLens-1M [10] dataset. Overall, *Random Search* and *Bayesian Hyper-opt* can generally find a good setting with less searching time than *Grid Search*. Therefore, users can choose the strategy for hyper-parameter tuning to strike a balance between accuracy and efficiency. Also, we implement the result visualization function, so that users can monitor the dynamic tuning process. Furthermore, by introducing the framework Ray [21] for scaling AI and Python applications, we can support parallel tuning with multiple GPUs for more efficient parameter search, and users can easily switch between parallel and serial tuning via different configurations. The usage examples of the two hyper-tuning tools are shown in Figure 5.

```
python significance_test.py --model_ours=BPR --
    model_baseline=NeuMF --dataset=ml-100k --
    run_times=30 --config_files="BPR.yaml NeuMF.yaml"
> recall@10: statistic=-21.593406593406602, pvalue
    =0.014730543885695793
> ndcg@10: statistic=-26.763440860215045, pvalue
    =0.011887928803273138
```

Figure 6: Usage example of significance test.

```
#running command
python run_recbole_group.py --model_list=NGCF,LightGCN
     --dataset=ml-100k --config_files=config.yaml --
    valid_latex=./latex/valid.tex --test_latex=./
    latex/test.tex

#latex-code generation
\begin{table}
\caption{Result Table}
\label{Result Table}
\begin{tabular}{ccccc}
\toprule
Model & recall@10 & mrr@10 & ndcg@10 & precision@10 \\
\midrule
NGCF & 0.2546 & 0.4991 & 0.3030 & 0.2049 \\
LightGCN & \bfseries 0.2571 & \bfseries 0.5071 & \
    bfseries 0.3038 & \bfseries 0.2055 \\
\bottomrule
\end{tabular}
\end{table}
```

Figure 7: Usage example of LaTeX-code generation.

2.2.4 Academic Performance Comparison. To facilitate academic research, an important role of our benchmark library is to provide a reliable performance comparison between models. For this purpose, we further implement scientific significance test in our framework to ensure the reliability of the comparison. Moreover, we develop a convenient built-in function to automatically generate result comparisons of different models in LaTeX format.

Scientific significance test. To conduct rigorous experiments, *significance test* is widely used in the research of recommender systems, which is a statistical procedure examining whether the performance comparison between models is significant or occasional. While, such a test mechanism has been seldom included in existing recommendation libraries. In this update, we add a new function to implement significance test for comparing various metrics of two specified models. Specifically, we use one-tailed paired t-test, *i.e.,* null hypothesis assumes that a target model cannot perform as well as the baseline. The statistical significance level is determined by users, with two widely used options of 0.01 and 0.05. As shown in Figure 6, we compare the BPR [26] model and NeuMF [12] model in 30 runs on MovieLens-100K dataset, and the results of significance test are presented under the command. With the calculated p-value, we can compare the two models on different metrics (*e.g.,* Recall and NDCG) in a more reliable way.

Table 6: Statistics of datasets for four recommendation tasks with benchmark configurations.

Tasks	Benchmark Datasets		
General	MovieLens-1M (6K users | 3K items)	Amazon-Books (40K users | 31K items)	Yelp (72K users | 43K items)
Sequential			
Context-aware	MovieLens-1M (7 fields | 10K features)	Avazu (23 fields | 2M features)	Criteo (39 fields | 4M features)
Knowledge-aware	MovieLens-1M (10K entities | 54 relations)	Amazon-Books (10K entities | 22 relations)	LastFM (94K entities | 12 relations)

Convenient LaTeX-format results. Since RecBole is mainly for research use, we further provide a new function that allows users to directly generate LaTeX-format results after running the toolkit, which can largely reduce efforts to manually construct tables. Specially, multiple models can be invoked in a single configuration file for performance comparison, and our library automatically converts the results into the LaTeX code. In this way, users can simply copy the output code from RecBole and the target table can be compiled in the LaTeX format. Figure 7 presents an example of the LaTeX-format results. In this example, we run NGCF [38] and LightGCN [11] (more methods can be included) in unified configurations on MovieLens-100K [10] dataset, and the results are automatically converted into LaTeX-code format. Then, the table for performance comparison can be compiled directly from the generated LaTeX code, which improves the efficiency of scientific writing.

2.3 Benchmark Configurations

RecBole has provided researchers with a standardized framework to reproduce baselines and design new models. However, as commented in the feedback of our users, it is still time-consuming to conduct hyper-parameter tuning for deriving decent performance. To facilitate researchers to reproduce model results, our team has sought to make the model implementations adhere to the original paper and source code (if any), and perform a rigorous evaluation on all models by detailed hyper-parameter search. In this update, we release the hyper-parameter tuning range and recommended parameter configurations for each model on three benchmark datasets respectively. According to the hyper-parameter settings and configurations in https://recbole.io/hyperparameters, researchers can conveniently obtain the model results on benchmark datasets, which greatly reduces the time cost for hyper-parameter tuning. Next, we introduce our efforts in reproducing the benchmark results.

• **Comprehensive models**. For the four types of recommendation tasks, we release the recommended hyper-parameter tuning settings on three benchmark datasets (three for each task), including general, sequential, context-aware and knowledge-aware recommendation. Meantime, our team encourages the community contributors to work together for maintaining and updating the benchmark configurations (especially the original authors). As future works, we also consider publishing the hyper-parameter settings of the sub-packages in RecBole 2.0 [46] to facilitate follow-up research.

• **Representative datasets**. For each type of the four recommendation tasks in RecBole, we provide three benchmark datasets respectively. For *general* and *sequential recommendation*, we adopt classic movie recommendation dataset MovieLens-1M [10], the latest Yelp dataset [33] in 2022 and Amazon Books dataset [19] in 2018, considering the popularity, domains and scales. For *context-aware recommendation*, we also adopt MovieLens-1M [10] since it contains attribute information for movies. Besides, Criteo [5] and Avazu [24] are two frequently used datasets for click-through rate predictions, which are used as the benchmark datasets for context-aware recommendation. For *knowledge-aware recommendation*, we use MovieLens-1M [10], Amazon Books [19], and LastFM [27] from KB4Rec [45] as our benchmark datasets, which are all linked to Freebase and widely adopted by academia on knowledge-aware models [22, 35]. To ensure data quality, we only retain data with a rating not less than 3 as positive samples. Moreover, we employ 10-core filtering for users and items to remove inactive users and unpopular items. In addition, 5-core filtering is also applied to the nodes and relations in the knowledge-aware models. As for data splitting, we utilize the *leave-one-out* strategy in temporal order for sequential models, while we split data into train/validation/test sets with a ratio of 8:1:1 [44] for the other three types of models. Table 6 shows the statistics of all processed benchmark datasets.

• **Evaluation metrics**. In the hyper-parameter tuning process, we mainly use two types of evaluation metrics, *i.e.,* value-based metrics and ranking-based metrics. For the context-aware recommendation, we adopt two popular value-based metrics, *i.e.,* AUC and Logloss, and AUC is used as an indicator to apply early-stopping strategy. To evaluate the other three types of models, we adopt widely used ranking-based metrics, including Recall@K, Precision@K, HR@K, NDCG@K, and MRR@K, where K is set to 10, and NDCG@10 is utilized for early stopping. Note that for all ranking-based metrics, we use full-ranking evaluation rather than sampling-based evaluation to improve the reliability, since recent studies [47] have found that different sampling numbers of sampling-based evaluation may lead to inconsistent conclusions.

• **Fine-grained hyper-parameter tuning**. To obtain the optimal (or nearly optimal) performance for a recommendation model, we utilize the grid search strategy for parameter tuning on each dataset. As for the search range of hyper-parameters, we try to follow the settings of original papers and fine-tune specific parameters according to the original training methods. But on some large datasets, due to time consumption, we will appropriately reduce the parameter range according to the search results on smaller datasets.

• **Reproducible and fair configurations**. To ensure a fair comparison, the common parameters and configurations between different models should be kept as consistent as possible. In context-aware models, we set the embedding size of each feature field to 16, while the embedding size in the other three types of models is set to 64. The training batch size of all models on the same dataset is set to the same value. And we optimize all methods with

Adam [14] optimizer and initialize them with the default Xavier distribution. For sampling-based models, we all utilize uniform negative sampling and sample one negative sampling for each interacted one. Apart from the above parameters and configurations, we also release the detailed configuration files of datasets and models, the hyper-parameter tuning range and recommended model parameters. Based on these public configurations, researchers can reproduce the benchmark results of different models conveniently.

Note that in this update, we do not directly release the benchmark results, but instead report the hyper-parameter search range and the recommended parameter configurations for reproducibility. It allows more space to further specification when more resources or efforts can be employed. Since no library can enumerate the complete search space of parameters, we attempt to derive a good parameter configuration for decent performance instead of optimal results, which can be further optimized for specific applications.

3 USER-FRIENDLY USAGE INSTRUCTIONS

As a user-friendly recommendation library, RecBole builds a systematic open-source environment, which maintains the project website, code and dataset repositories, reference papers and API documentation. In this update, we further improve the user experience from the following two key aspects, *i.e., detailed documentation* and *well-organized frequently asked questions* (FAQ).

3.1 Detailed Documentation

In order to facilitate researchers to use our library, we have been updating the documentation all the time. After the release of RecBole 2.0 [46] and the new update in this paper, we have accordingly upgraded the documentation to improve the sustainability. In particular, we enrich our website by officially including RecBole 2.0 and benchmark configurations, and provide more practical tutorials for newcomers to get started quickly with our new features.

• **Comprehensive website**. For the website (https://recbole.io), besides the web page of benchmark configurations in Section 2.3, we add another page that provides a brief overview of all sub-packages in RecBole 2.0 [46]. We also demonstrate the main features of each sub-package with usage examples to facilitate a quick start. Furthermore, we summarize all the implemented models based on RecBole in the model page. Through the built-in hyperlink and retrieval function, users can take an overall view of more than 100 models and quickly jump to the corresponding implementation.

• **Convenient tutorials**. To further improve the user experience, we update the tutorials of RecBole by including detailed illustrations for both basic usage and new features. To help new users quickly get started with our project, we provide step-by-step instructions for running each of the four types of models. We also consider the usage demands of advanced users and add instructions for customized training strategies to facilitate self-defined trainers. Moreover, we provide detailed usage examples to demonstrate the utilization of newly developed features including distributed computing, mixed precision training, significance test and parallel parameter tuning.

3.2 Well-organized FAQ

Ever since the first release of RecBole, our team has been devoted to creating an interactive community environment for recommender systems, and has been continuously solving incoming issues and answering the comments on GitHub projects. So far, in RecBole, we have solved 604 issues and 71 discussions. These issues provide valuable experience for new use of our library, which involves many key aspects of the open-source library for recommendation. Thus, we organize the frequently asked questions in a more accessible way at the link: https://github.com/RUCAIBox/RecBole/issues/1701, which avoids duplicate issues and further improves user experience.

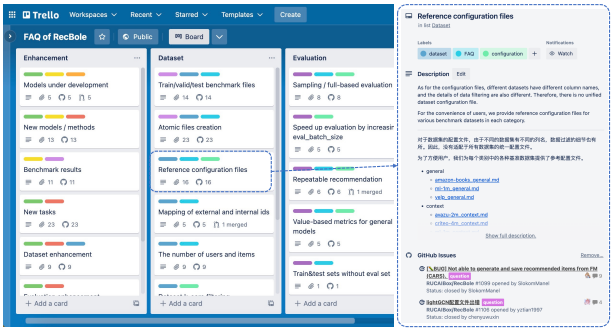

Figure 8: The screenshot of the FAQ page.

With this FAQ page, users can browse the enhancement suggestions, use problems, academic discussions and experimental guidance at a time, based on a two-year open-source development. For each included question, we not only provide the official answer with detailed explanations, but also link this problem to its related issues, pull requests and discussions. By doing this, users can get important advice and guidance from the experience of previous users. Besides, the related pull requests also include the technical details of implementation for a better understanding of the question.

4 DEVELOPMENT GUIDELINE

In recent years, many recommendation libraries have been developed [25, 28, 32, 48]. Despite the progress, the teams of these libraries typically have different development focuses, *e.g.,* unified code framework or comprehensive support for evaluation. However, these libraries (including previous RecBole) have not explicitly investigated the needs of real users for an open-source toolkit. For this purpose, we aim to analyze and derive user requirements from two aspects, *i.e., historical issues* and *investigation questionnaires.*

Analysis with GitHub issues of existing libraries. As the first kind of resources, we consider collecting the GitHub issues of existing libraries to analyze user needs with recommendation toolkits, since users would ask, discuss and suggest what they think are important on the use of open-source libraries. Not limited to our library, we select other popular libraries such as DeepCTR [28], TorchRec [25] and EasyRec [32], for issue collection. After collecting the text corpus, we adopt the pre-trained BERTopic [8] to analyze and extract the concerned topics (*i.e.,* key phrases) of users. The topics are manually grouped into five major categories as shown in Table 7. For each category, we manually generate four aspect labels that can well cover the mined topics. Afterward, we score the importance of each topic based on the proportion of issues involved

Table 7: Investigation of user needs for open-source recommendation libraries and future works of RecBole. We measure the importance of user requirements in five aspects from both GitHub issues (Issue) and user questionnaires (Ques.). The most concerned aspects described in questionnaires and issues are marked in *bold* and *underlined* fonts, respectively. The source data of our collected questionnaires can be found at https://github.com/RUCAIBox/RecBole/asset/questionnaire.xlsx.

	Aspect	Issue	Ques.	Past Efforts in RecBole	Future Plans for RecBole
Dataset	**Quantity and coverage**	3.43	4.42	41 datasets covering frequently-used datasets	More public datasets
	Novelty and recency	2.95	3.99	Novel public datasets	Periodic update for the latest datasets
	Flexibility and customization	5.00	4.42	Formatted atomic files and data flow	User-defined data input and processing
	Diversity of types	3.67	4.28	Multiple user/item attributes and 3 knowledge-aware datasets	Datasets with diversified side information
Model	Quantity and coverage	4.09	4.29	4 tasks and 86 models	More tasks and models
	Novelty and recency	3.19	4.22	Continuously adding new models	More tasks involved
	Efficiency and memory utilization	4.54	4.29	Multi-GPU utilization	Optimizing the usage of CPU and GPU
	Flexibility and customization	5.00	4.42	Customized trainer and recommender	More custom interfaces
	Universality in model design	2.91	4.39	Unified pipeline and framework	Unify network structure and API
	Explanation of configurations	4.09	4.41	Well-organized documentation	Detailed examples of configurations
Metric	**Quantity and coverage**	2.78	4.27	16 metrics for ranking and value predictions	Beyond-accuracy metrics such as fairness
	Novelty and recency	1.11	3.80	Classic evaluation metrics	Up-to-date metric involved
	Flexibility and Customization	5.00	4.15	Customized metric for users	More flexible design for metrics
Evaluation	**Comprehensive benchmark**	5.00	4.52	Benchmark hyper-parameter configurations	More benchmark datasets involved
	Conformity to original papers	2.50	4.27	Adhering to the original paper and code	Double-review check of implementation
	Case study	1.67	3.91	Usage example of case study	More practical examples for all tasks
	Significance test	0.83	3.80	Interface for significance test	More scientific comparison and analysis
	Hyper-parameter tuning	2.59	3.96	Serial hyper-opt and parallel Ray	Improving the efficiency
Usability	**Expansibility**	5.00	4.56	Including 8 sub-packages in RecBole 2.0	More expanded projects
	Documentation	3.67	4.53	Detailed instructions with FAQ	Examples of executable .ipynb
	Code readability	4.32	4.56	Formatting Python code according to PEP8	Code interpretation in documents
	Friendly interaction	2.43	4.29	Well-maintained issues and discussions	More timely communication and feedback

in the corpus, and then further normalize it to a real-value score within the range $[0, 5]$ (displayed in the third column of Table 7).

Analysis with user questionnaires. Since the collected GitHub issues mainly reflect the experiences of users for existing libraries, we further consider extending the audience for collecting more suggestions to improve recommendation libraries. Based on the derived categories and key phrases in Table 7, we design a questionnaire to acquire user requirements on open-source recommendation library. We distribute questionnaires mainly for researchers in the field of recommender systems, and collect 153 valid (without missing slots) feedbacks in total. For each aspect in a category, we average the ratings (*i.e.,* in the five-star format) from all participants as the importance score (displayed in the fourth column of Table 7).

Findings and guidelines. The two kinds of resources provide user requirements on recommendation libraries in different perspectives, and we can make a comparison about their focuses based on the importance scores. Interestingly, these two perspectives seem to be well aligned on some common aspects (in both bold and underlined fonts). From Table 7, we can easily identify the most important aspects by category: *flexibility & customization* (dataset and model), *comprehensiveness* (evaluation), and *explanability* (usability). While, for metrics, questionnaire users seem to focus on *flexibility & customization* instead of *coverage* as GitHub users. A possible reason is that the libraries for collecting issues well support evaluation protocols, and users typically have fewer issues in this aspect. Furthermore, we list the efforts that RecBole has included in past versions and also the future plans. These findings are also useful to guide the development of other recommendation libraries.

5 CONCLUSIONS

As a user-friendly open-source library, RecBole is expected to help researchers easily reproduce baselines as well as develop new algorithms, serving as a comprehensive benchmark library for recommendation. Since our release in 2020, RecBole has attracted extensive attention from open-source contributors on GitHub, with over 2500 stars and 490 forks. As recommender systems have been evolving rapidly, we have received a number of requests and suggestions. This motivates us to make some significant improvements on our library, so as to fulfill the user requirements and benefit the research community. Therefore, we release this update for making RecBole more user-friendly and easy-to-use as a benchmark library. Our update highlights can be featured in four aspects, *i.e.,* more flexible data processing, more efficient training and tuning, more reproducible configurations and more user-friendly documentation. Finally, we provide guidance for the development of open-source recommendation library through the analysis of historical feedback and constructive questionnaires. We believe this update is an important extension to RecBole, which is an invaluable resource to the research community. In the future, we will continuously update RecBole to make it more *flexible*, *up-to-date* and *user-friendly*.

ACKNOWLEDGEMENTS

This work was partially supported by the National Natural Science Foundation of China under Grant No. 62222215 and 62102420, Beijing Natural Science Foundation under Grant No. 4222027, and Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098. We sincerely thank the support from all users and previous developing members of RecBole.

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* 11, 9 (2018), 137.

[2] Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. 2017. A neural collaborative filtering model with interaction-based neighborhood. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1979–1982.

[3] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[4] Hugo Caselles-Dupré, Florian Lesaint, and Jimena Royo-Letelier. 2018. Word2vec applied to recommendation: Hyperparameters matter. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 352–356.

[5] Kaggle Display Advertising Challenge. 2014. Predict click-through rates on display ads. https://www.kaggle.com/c/criteo-display-ad-challenge/data. (2014).

[6] Xinyan Fan, Zheng Liu, Jianxun Lian, Wayne Xin Zhao, Xing Xie, and Ji-Rong Wen. 2021. Lighter and better: low-rank decomposed self-attention networks for next-item recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1733–1737.

[7] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems (TOIS)* 39, 2 (2021), 1–49.

[8] Maarten Grootendorst. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794* (2022).

[9] Huifeng Guo, Bo Chen, Ruiming Tang, Weinan Zhang, Zhenguo Li, and Xiuqiang He. 2021. An embedding learning framework for numerical features in ctr prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2910–2918.

[10] F Maxwell Harper. 2015. The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis) 5 4 (2015) 1–19. F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis) 5 4 (2015) 1–19.

[11] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[12] 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*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182.

[13] Yupeng Hou, Binbin Hu, Zhiqiang Zhang, and Wayne Xin Zhao. 2022. Core: simple and effective session-based recommendation within consistent representation space. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1796–1801.

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 539–548.

[16] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the ACM Web Conference 2022*. 2320–2329.

[17] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A Simple and Strong Baseline for Collaborative Filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1243–1252.

[18] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S Vetter. 2018. Nvidia tensor core programmability, performance & precision. In *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*. IEEE, 522–531.

[19] Julian McAuley. 2014. The Amazon Dataset. http://jmcauley.ucsd.edu/data/amazon/. (2014).

[20] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740* (2017).

[21] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. 2018. Ray: A distributed framework for emerging AI applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 561–577.

[22] Shanlei Mu, Yaliang Li, Wayne Xin Zhao, Siqing Li, and Ji-Rong Wen. 2021. Knowledge-guided disentangled representation learning for recommender systems. *ACM Transactions on Information Systems (TOIS)* 40, 1 (2021), 1–26.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[24] Kaggle Click-Through Rate Prediction. 2014. Predict whether a mobile ad will be clicked. https://www.kaggle.com/c/avazu-ctr-prediction/data. (2014).

[25] PyTorch. 2022. TorchRec: A Pytorch domain library for recommendation systems. https://github.com/pytorch/torchrec. (2022).

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.

[27] Markus Schedl. 2016. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on international conference on multimedia retrieval*. 103–110.

[28] Weichen Shen. 2017. DeepCTR: Easy-to-use,Modular and Extendible package of deep-learning based CTR models. https://github.com/shenweichen/deepctr.

[29] J. Snoek, H. Larochelle, and R. P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in neural information processing systems* 4 (2012).

[30] Harald Steck, Maria Dimakopoulou, Nickolai Riabov, and Tony Jebara. 2020. Admm slim: Sparse recommendations for many users. In *Proceedings of the 13th international conference on web search and data mining*. 555–563.

[31] Qiaoyu Tan, Jianwei Zhang, Jiangchao Yao, Ninghao Liu, Jingren Zhou, Hongxia Yang, and Xia Hu. 2021. Sparse-interest network for sequential recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 598–606.

[32] EasyRec Team. 2022. EasyRec: A tensorflow framework for large scale recommendation algorithms. https://github.com/alibaba/EasyRec. (2022).

[33] Yelp Team. 2018. The Yelp Dataset. https://www.yelp.com/dataset. (2018).

[34] Zhen Tian, Ting Bai, Zibin Zhang, Zhiyuan Xu, Kangyi Lin, Ji-Rong Wen, and Wayne Xin Zhao. 2023. Directed Acyclic Graph Factorization Machines for CTR Prediction via Knowledge Distillation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 715–723.

[35] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, ShaoZhang Niu, and Jimmy Huang. 2020. KERL: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 209–218.

[36] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*. 1785–1797.

[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 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 950–958.

[38] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[39] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the Web Conference 2021*. 878–887.

[40] 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 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 135–144.

[41] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 726–735.

[42] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 1259–1273.

[43] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 785–788.

[44] Wayne Xin Zhao, Junhua Chen, Pengfei Wang, Qi Gu, and Ji-Rong Wen. 2020. Revisiting alternative experimental settings for evaluating top-n item recommendation algorithms. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2329–2332.

[45] Wayne Xin Zhao, Gaole He, Kunlin Yang, Hongjian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. 2019. Kb4rec: A data set for linking knowledge bases with recommender systems. *Data Intelligence* 1, 2 (2019), 121–136.

[46] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, et al. 2022. RecBole 2.0: Towards a More Up-to-Date Recommendation Library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4722–4726.

[47] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. 2022. A revisiting study of appropriate offline evaluation for top-N recommendation algorithms. *ACM Transactions on Information Systems* 41, 2 (2022), 1–41.

[48] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4653–4664.

[49] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1893–1902.

[50] Ding Zou, Wei Wei, Xian-Ling Mao, Ziyang Wang, Minghui Qiu, Feida Zhu, and Xin Cao. 2022. Multi-level cross-view contrastive learning for knowledge-aware recommender system. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1358–1368.