BARS: Towards Open Benchmarking for Recommender Systems

Jieming Zhu[§]
Huawei Noah's Ark Lab
Shenzhen, China
jiemingzhu@ieee.org

Quanyu Dai
Huawei Noah's Ark Lab
Shenzhen, China

Liangcai Su
Tsinghua University
Shenzhen, China

Rong Ma
Tsinghua University
Beijing, China

Jinyang Liu[★]
The Chinese University of Hong Kong
Hong Kong, China

Guohao Cai
Huawei Noah's Ark Lab
Shenzhen, China

Xi Xiao[§]
Tsinghua University
Shenzhen, China
xiaox@sz.tsinghua.edu.cn

Rui Zhang
ruizhang.info
rayteam@yeah.net

ABSTRACT

The past two decades have witnessed the rapid development of personalized recommendation techniques. Despite significant progress made in both research and practice of recommender systems, to date, there is a lack of a widely-recognized benchmarking standard in this field. Many existing studies perform model evaluations and comparisons in an ad-hoc manner, for example, by employing their own private data splits or using different experimental settings. Such conventions not only increase the difficulty in reproducing existing studies, but also lead to inconsistent experimental results among them. This largely limits the credibility and practical value of research results in this field. To tackle these issues, we present an initiative project (namely BARS) aiming for open benchmarking for recommender systems. In comparison to some earlier attempts towards this goal, we take a further step by setting up a standardized benchmarking pipeline for reproducible research, which integrates all the details about datasets, source code, hyper-parameter settings, running logs, and evaluation results. The benchmark is designed with comprehensiveness and sustainability in mind. It covers both matching and ranking tasks, and also enables researchers to easily follow and contribute to the research in this field. This project will not only reduce the redundant efforts of researchers to re-implement or re-run existing baselines, but also drive more solid and reproducible research on recommender systems. We would like to call upon everyone to use the BARS benchmark for future evaluation, and contribute to the project through the portal at: https://openbenchmark.github.io/BARS.

[★] Part of the work was done when the author studied at Sun Yat-Sen University.
[§] Corresponding authors.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**; **Computational advertising**.

KEYWORDS

Recommender systems; benchmarking; CTR prediction; item matching, collaborative filtering

1 INTRODUCTION

Personalized recommendation plays an indispensable role in our daily life to help people discover information of their interests. It has been adopted in a wide range of online applications, such as E-commerce, social media, news feeds, and music streaming. The application of recommender systems not only helps users alleviate information overload but also serves as an important source for platform vendors to gain revenues from user interactions (e.g., clicks, downloads, or purchases). Typically, modern recommender systems in industry consist of two major phases, namely matching and ranking [27]. The matching phase (*a.k.a*, retrieval phase) aims for efficient retrieval of candidate items and helps reduce the candidate item set from millions to hundreds while guaranteeing a high recall rate. The ranking phase further ranks the retrieved items by leveraging abundant user/item/context features and returns the top-k (*e.g.*, in tens) items to users.

Under the wave of deep learning, we have witnessed the rapid development of recommendation techniques in both phases. Many deep neural networks, such as auto-encoders [56, 63, 89], graph neural networks (GNN) [40, 52, 54, 96], convolution neural networks [57], and transformers [92, 95], have been applied to recommender systems for better matching and ranking capabilities. Moreover, numerous new recommendation algorithms are published every year. Despite the widespread research of recommender

systems, there is still a lack of a widely-recognized benchmarking standard in this field. This raises a growing concern that a significant number of research papers lack rigor in evaluation and reproducibility of reported results [28, 47, 98]. In the literature, existing studies on recommender systems are often evaluated in an ad-hoc way, for example, by employing their own private data splits, using a different experimental setting, or adopting a weak baseline (e.g., not well tuned). In many cases, the reported results cannot be easily reproduced due to the lack of either data preprocessing details, model implementations, hyper-parameter configurations, or even all of them. This not only leads to inconsistent results among existing studies but also makes fairly comparing model effectiveness a non-trivial task, which largely limits the credibility and practical value of research results in this field [44]. Recently, the critical issue has attracted more and more attention and has been reported in several pioneer studies [28, 29, 80, 83, 98].

Inspired by the success of the ImageNet benchmark [32] in the CV domain and the GLUE benchmark [102] in the NLP domain, we propose an initiative for benchmarking recommender systems related research in a standard and reproducible manner. Many precious research efforts have been made towards this goal, which can be broadly categorized into three groups: *1) Benchmarking datasets*. A lot of real-world benchmarking datasets have been released to promote research for recommender systems, including MovieLens [11], Yelp [6], AmazonBooks [1], Gowalla [4], Criteo [3], just to name a few. Furthermore, a large set of preprocessed datasets are readily available in RecBole [125]. *2) Benchmarking tools*. With the evolution of deep learning-based recommendation techniques, many useful open-source libraries and tools have been developed, such as DeepRec [123], daisyRec [98], NeuRec [111], RecBole [125], DeepCTR [7], FuxiCTR [9], TensorFlow Recommenders [14], TorchRec [15], PaddleRec [13], and EasyRec [8], each providing tens of popular recommendation algorithms[1]. *3) Benchmarking results*. Some recent work performs evaluation studies to reproduce existing methods or identify their non-reproducibility issues in deep learning-based recommendation [98], session-based recommendation [60], and review-based recommendation [83]. However, none of them provides a comprehensive leaderboard of benchmarking results with high reproducibility. In this work, we aim to chain the above pieces of work together (i.e., datasets, tools, and results) to form a standard open benchmarking pipeline for recommender systems, namely BARS.

Two closely-related studies to ours are Elliot [17] and Recbole [125], which provide rigorous evaluation frameworks for developing and benchmarking recommendation models. In contrast to some earlier attempts [51, 84], the design of our benchmarking system has the following merits.

- **Standardization**: We propose a standardized benchmarking pipeline to promote reproducible research, which involves seven necessary artifacts. In addition to the widely-existing open datasets and open-source models, we also mark data splits, evaluation

protocols, hyper-parameter configurations, running logs, evaluation results as required artifacts, which need to be properly recorded during benchmarking. The standardized pipeline not only encourages reuse of open datasets as well as open-source code, but also aims for tractable data splits and reproducible evaluation results. Without such a standard, existing studies may only deliver parts of the artifacts (e.g., only code in DeepCTR [7], lack of hyper-parameters and running logs in daisyRec [98]).

- **Reproducibility**: Reproducibility is the first-class goal of our benchmark. For each result in BARS, we provide the corresponding reproducing steps in details (along with the required artifacts) to allow anyone for easy reproduction, including hardware and software environment, dataset and model configurations, as well as running scripts and logs. These could also serve as good guidelines for junior researchers.

- **Reusability**: On one hand, none of the datasets were created from scratch for the benchmark. We reuse pre-existing datasets (e.g., Criteo [3]) because they have been implicitly agreed upon by the community due to their wide use in existing papers. We also reuse the open-source code wherever possible (e.g., daisyRec [98]) to encourage the contributions from different parties. On the other hand, our work aims to provide well-tuned benchmarking results and settings that could be directly reused for future research. Concretely, we evaluate the performance of tens of state-of-the-art (SOTA) models and provide strong baselines to compare with. Given our benchmarking results, researchers could easily gauge the effectiveness of new models, while largely reduce the tedious yet redundant efforts to re-implement and re-run the baselines for publishing a new paper.

- **Comprehensiveness**: To aim for comprehensiveness of our BARS benchmark, we have integrated over 6 preprocessed datasets and more than 70 recommendation algorithms covering both matching and ranking phases. The thorough benchmarking results on these models and datasets provide a systematic view of the effectiveness of current recommendation algorithms and facilitate better understanding of the real progress that we have made. To the best of our knowledge, our leaderboard provides the most comprehensive benchmarking results in the recommender systems community.

- **Sustainability**: A benchmark will become stale soon if it is not regularly maintained and widely recognized by the community. To ensure sustainability of our benchmark, we build a benchmark website like GLUE [102] that allows anyone in the community to easily follow and contribute. We call for contributions from any aspect to foster sustainable development of BARS, including extending new datasets, benchmarking existing models on new data splits, implementing new SOTA models, reporting improved results with better hyper-parameter settings, polishing the documentation, etc.

- **Industrial-level**: Existing studies [28, 98] focus primarily on conventional recommendation algorithms, such as collaborative filtering (CF), which fail to distinguish between matching and ranking tasks from an industrial view. Actually, CF is only one type of matching methods used in industrial recommender systems. Instead, our work explicitly classifies existing recommendation methods into matching and ranking phases, and performs benchmarking with different metrics on candidate item matching

---

[1]It is worth mentioning that sometimes benchmarking datasets or tools are also referred to as "benchmarks" in the literature, since they provide necessary data and components (e.g., data loader, metric evaluator, hyperparameter tuner, etc.) for benchmarking use. Yet, we target at a GLUE-like leaderboard as the benchmark in this paper, and refer to the process to obtain such a leaderboard as the benchmarking pipeline.

and click-through rate (CTR) prediction tasks, respectively. Thus, our benchmark better fits the need for industrial use.

Bearing these design goals in mind, we build the BARS benchmark, which has been released at https://openbenchmark.github.io/BARS. While the ultimate goal of BARS is to drive open benchmarking for all recommendation tasks, as an initiative work, our current version covers benchmarking results for candidate item matching and CTR prediction. In particular, we ran over 8,000 experiments for more than 15,000 GPU hours (~625 days) in a standard setting to benchmark more than 70 recommendation models on six widely-used datasets (More results are available on the BARS website). The large number of experiments ensure sufficient hyper-parameter tuning of each model via grid search. Under sufficient model tuning, we found that the effectiveness of many recent methods may be exaggerated. Many of them have smaller differences than expected and sometimes are even inconsistent with what reported in the literature. We believe that easy access to consistently split and pre-processed datasets with reusable baseline results could help the research community avoid reporting inconsistent or misleading results in their future work.

In summary, our work could serve as a useful resource for several different group of readers: *Researchers* are easy to get a full view of model improvements made over recent years and also gauge the effectiveness of new models conveniently. *Practitioners* could reduce unnecessary trials by comparing different models via the benchmark leadboard. *Competitors* can easily implement baseline solutions by leveraging our source code and benchmarking scripts. *Educators* could use BARS as a valuable resource to teach and study recommendation algorithms.

By setting up an open benchmarking leaderboard, together with the freely available benchmarking artifacts (e.g., datasets, code, configurations, results, and reproducing steps), we hope that the BARS project could benefit all researchers, practitioners, and educators in the community. We also call for generous contributions from the whole community to improve this open benchmarking project and to keep healthy development with the rapid evolution of recommender systems research.

We emphasize that this is not the first and will not be the last work on benchmarking recommender systems. There are a large body of precious research work (e.g., [17, 28, 98, 125], just to name a few) that has been done towards this goal. In our previous work [130], we have built a benchmark specifically for the CTR prediction task. In this work, we initiate the effort of creating an open benchmarking standard for various recommendation tasks in both matching and ranking phases and cover a larger number of models and datasets in the benchmark (detailed in Section 2.4). Yet, implementing and tuning models are very time-consuming and labor-intensive, thus still limiting the scope of the BARS benchmark. We thus would like to call for contributions from the community to build a widely-recognized benchmark for recommender systems.

2 BACKGROUND AND RELATED WORK

2.1 Overview of Recommender Systems

The goal of recommender systems is to recommend new items to users from a large number of candidate items. Figure 1 illustrates the overall workflow of modern recommender systems. It mainly
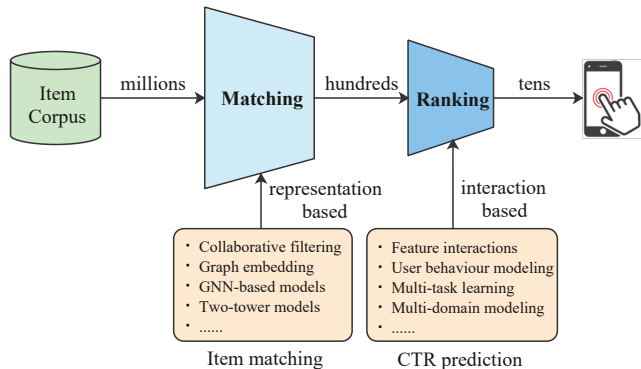

Figure 1: A simplified workflow of modern industrial RS.

contains two phases, i.e., matching and ranking. The matching phase aims to perform retrieval of candidate items from a large item corpus (with millions of items) with high efficiency, while the ranking phase targets at learning the fine-grained personalized ranking of the retrieved items with click-through rate (CTR) prediction[2] and returning a small number of top-ranking (i.e., in tens) items from the candidates. Through these two phases, the recommender system can strike a balance between accuracy and efficiency. In the following, we introduce the related work of these two recommendation phases.

2.2 Candidate Item Matching

In the matching phase, retrieval efficiency is critically important due to the extremely large item corpus in real systems. Thus, the majority of matching algorithms applies representation-based models that enable fast vector retrieval (e.g., via Faiss [45]) and only consider coarse features of users and items. They usually first obtain user and item representations and then compute user-item matching scores through simple similarity measures (e.g., inner product, cosine). The representative methods can be summarized into the several categories, including collaborative filtering (CF) [94], two-tower models [27, 87], graph embedding models [72, 99], graph neural networks (GNN) [40, 121], and autoencoder-based models [56, 113].

Collaborative filtering [94] methods leverage the collaborative information among users and items to predict users' preferences on candidate items. The classic CF models include neighbourhood-based ones [85] and matrix factorization (MF)-based ones [48, 65]. In recent years, various variants have also been proposed, such as NeuMF [48], ENMF [23], and MF-CCL [66].

The majority of CF methods, however, are focused on learning embeddings of user and item ids. To better incorporate user features and items features in representation learning, two-tower neural networks are often applied. DSSM [87] is a classic model in which users and items are modelled by two separate network branches. YoutubeDNN [27] and SimpleX [66] can be viewed as simplified variants of two-tower models, as only item ids are used in the item branch. Meanwhile, negative sampling techniques [33, 104, 117] are critical for training such two-tower models. These models ensure

---

[2]The ranking phase may also involve other similar types of item scoring, such as CVR prediction, "like" prediction, and reading/watching time prediction

high efficiency to fulfill the requirements of practical applications and thus have been widely adopted in industry.

To fully capture the collaborative signals, graph-based algorithms have been widely studied in recent years, including the graph embedding-based models [30, 31, 72] and GNN-based models [40, 107, 121]. The reason is that interaction data can be naturally modeled as a user-item bipartite graph, thus candidate item matching can be reformulated as a link prediction problem. Graph embedding models (e.g., DeepWalk [72], Node2Vec [36], and EGES [103]) usually first learn node representations to capture graph structural information, and then perform link prediction with the learned representations. Currently, GNN-based models, such as PinSage [121], LightGCN [40], and UltraGCN [67], demonstrate quite strong performance for candidate item matching due to their effectiveness in modeling graph data.

Last but not least, autoencoder-based models have also been widely studied for item matching, such as CDAE [113] based on denoising autoencoder, Mult-VAE [56] based on variational autoencoder, and MacridVAE [63] based on β-VAE.

While the aforementioned models aim to learn effective representations, how to leverage them for efficiently top-k item retrieval from a large item pool is also a critical problem for candidate item matching. For this purpose, many techniques and tools have been developed to support approximate nearest neighbor search (e.g., Faiss [45], Milvus [10], and Scann [38]).

2.3 CTR Prediction

The ranking phase aims for personalized ranking of a small set (usually hundreds) of candidate items by considering abundant features of users, items and contexts. CTR prediction is one of the most representative tasks for ranking. Due to the practical value of this task, a large number of research efforts have been made in this direction, and many effective models have been designed, ranging from logistic regression models [68, 81], to factorization machines [78], to deep neural networks (DNN) [25, 37].

To achieve accurate CTR prediction, feature interaction modeling plays an essential role [124]. A series of DNN-based models have achieved impressive results through designing various feature interaction operators, such as product operators in PNN [76], NFM [39], and DCN [105], convolutional operators in CCPM [58], FGCNN [57], and FiGNN [54], and attention operators in AFM [114], FiBiNET [43], and AutoInt [92]. Meanwhile, with the success of the wide and deep learning framework, two-stream networks (e.g., Wide&Deep [25], DeepFM [37], DCN [105], and AutoInt [92]) have been widely applied for CTR prediction tasks, which demonstrate better performance than single-stream networks (e.g., DNN [27]).

In addition, user behavior modeling serves as an important aspect in CTR prediction. To mine crucial patterns of user interests from historical behaviors, researchers have designed various types of models, ranging from attention-based models [110, 127, 128], to memory network based-models [73, 77], to retrieval-based models [74, 75]. More recently, multi-task learning [62, 64] and multi-domain learning [50, 90] models have also been widely studied to alleviate the data sparsity issue in CTR prediction. Some other important techniques include large-scale training [126], multi-modal fusion [116], model ensemble [129], and continual learning [22].

2.4 Benchmarking and Reproducibility for Recommender Systems

Recently, the problems of unreproducible and unfair comparison are raising more and more concerns from the community. A series of studies [16, 28, 80, 98] have made in-depth experiments and analysis to point out this serious problem in recommendation, and further call for establishing a unified benchmark.

To encourage more reproducible research, many respectable efforts have been devoted to designing tools or benchmarks. Table 1 presents a comparison of our BARS project against some representative existing work. We mainly compare them from two perspectives, including: 1) whether the work provides tools or datasets for benchmarking use; 2) whether comprehensive and reproducible benchmarking results are available. It turns out that most of the pioneering efforts focus mainly on building flexible tools for convenient evaluation and benchmarking. Yet, there is still a lack of standard benchmarking pipeline and the comprehensive benchmarking leaderboard.

Among these existing studies, daisyRec [98] highlights the importance of open benchmarking in recommender systems and makes considerable efforts towards this direction by reporting the benchmarking results on some classic recommendation models. Concurrently, RecBole [125] and Elliot [17] both present a comprehensive evaluation framework with open-source APIs (e.g., data splitting, data filtering, metrics evaluation, statistical tests, hyperparameter tuning) and RecBole serves tens of SOTA models for reproducible evaluations, which becomes increasingly popular in the community. However, their work fails to provide a benchmark leaderboard of benchmarking results with detailed reproducing steps.

Difference from existing benchmark on CTR prediction [130]: We have established a benchmark for CTR prediction in our previous work [130]. In comparison, this work aims to make a further step and set up a standard open benchmarking pipeline for a wide range of recommendation tasks in both matching and ranking phases. The pipeline and required artifacts for open benchmarking specified in Section 3 are new to the community and allow researchers to easily expand the benchmarks to more tasks (e.g., re-ranking, sequential recommendation) and vertical scenarios (e.g., news recommendation, music recommendation). They could provide good guidance for researchers and facilitate more rigorous and reproducible research. In detail, we have made the following new contributions. Firstly, our BARS project provides the most comprehensive benchmarking results, w.r.t. coverage of datasets (10 vs. 4 in [130]) and models (64 vs. 24 in [130]) with existing libraries. Specifically, we have integrated more than 70 well-known models covering both matching and ranking phases. Secondly, our work proposes a standard open benchmarking pipeline (including 7 key components) as presented in Figure 2, which covers different recommendation tasks. It provides reusable dataset splits, detailed evaluation protocols, convenient open-source models and APIs, complete training logs, and well-tuned benchmarking results for reliable reproducibility of recommendation models. Third, we have built the BARS website with interactive leaderboards and visualizations for disseminating the benchmarking results. This enables researchers to easily follow and contribute to the leaderboards.

Table 1: Comparison of existing work (✓ | – | × means totally | partially | not met, respectively). "Partially met" indicates incomplete or private availability.

Usage	TorchRec	DeepCTR	EasyRec	DeepRec	daisyRec	NeuRec	RecBole	FuxiCTR	BARS
Benchmarking tools/datasets	✓	✓	✓	✓	✓	✓	✓	✓	–
Benchmarking results	×	×	×	×	–	×	–	×	✓
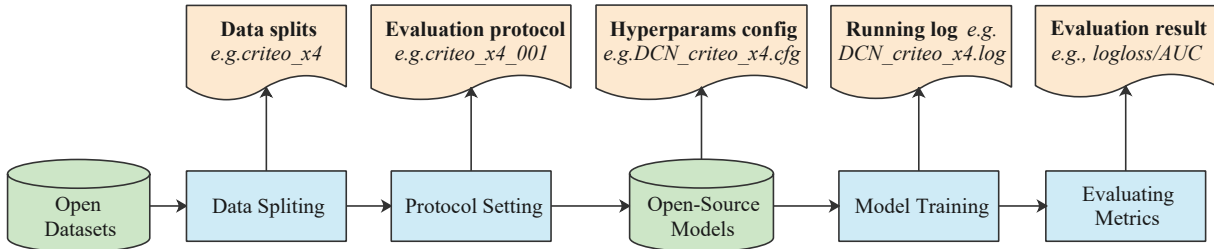
Figure 2: An open benchmarking pipeline for reproducible recommendation research.

3 OPEN BENCHMARKING PIPLELINE

Recommender systems have been an active research area for the past two decades. Unfortunately, there is still a lack of standardized benchmarking and evaluation protocols, which leads to many un-reproducbile issues in current studies. Openness in science builds the core for excellence in evidence-based research. Towards the goal for open science, we decide to set up a standardized open benchmarking pipeline for reproducible research in recommender systems. Figure 2 presents the open benchmarking pipeline for recommender systems. In summary, the pipeline consists of the following key components.

1) Open datasets: Open data are valuable to foster research progress. Fortunately, there have already been a large amount of publicly-available research datasets for recommendation tasks. Yet, most of them are at a small scale, e.g., less than millions of inter-actions between tens of thousands of users and items. Such data scale differs from the real-world problems in industry. Thus, we en-courage researchers to more often choose industrial-scale datasets, which would facilitate to build more practically applicable models and help reduce the gap between research and practice. This is also the goal of this benchmarking work.

2) Data splitting: While many possible datasets exist, their uses in the literature are arbitrary, which leads to incomparable and inconsistent results among existing papers. This is largely due to the lack of standardized data splitting and preprocessing. For model evaluation, in many cases, random data splitting is usually performed for training, validation, and testing. Most studies use their own data splits, but no details (neither the random seed nor the preprocessed data) are open, making it difficult for others to reuse the data. Recently, the authors of NGCF [107], AutoInt [92] and AFN [26] have shared the data processing scripts or the prepro-cessed data to follow-up researchers, demonstrating good practices for reproducibile research. Our work aims to establish such good practices on data splitting. We assign a unique dataset ID for each specific data split, e.g., Criteo_x4 denote the unique splits in [92].

This could not only facilitate the data reuse in the community ac-cording to dataset IDs, but also allow the research results using the same dataset ID directly comparable.

3) Setting evaluation protocol: For each specific data split, we define some specific evaluation protocols to allow for fair compar-isons among different models. For example, in Criteo_x4, we follow the setting in AutoInt [92] and set *min_counts*=10 to filter rare fea-tures and feature embedding dimension d=16. When a different evaluation protocol (feature embedding d=40 as in FGCNN [57]) is applied, we denote it with a new ID (e.g., Criteo_x4_002). Record-ing such a unique evaluation protocol would allow the reuse of all the benchmarking results for consistent comparisons in future research.

4) Open-source models: In the spirit of open science, many pa-pers have open sourced their model implementations. Even though the official source code may be missing, third-party implemen-tations often exist for an effective model. In many cases, these open-source models could be found at http://paperswithcode.com. However, despite the availability of model code, the details of hyper-parameter configurations may still be missing or incomplete. This reduce reproducibility because it is not inappropriate to directly run the model with default hyper-parameters. Moreover, most current papers only release their own model code, but not along with their baseline implementations and settings, which largely results in comparisons to weak baselines and thus inconsistent performance increases [119]. To promote reproducible research, our benchmark-ing work aims to record detailed hyper-parameter configurations for each experiment and demonstrate the reproducing steps.

5) Model training: When training a model, it is beneficial to record the training details into the log, which could help others to quickly glance over the running process of the model. In our work, we advocate to output the running log along with the running environments (e.g., hardware requirements, software versions) as necessary benchmarking artifacts.

6) Evaluating metrics: Evaluation metrics are key to measure model performance and make comparisons. Yet, it is not uncommon

that different metrics are used in different papers for evaluation. To standardize the metrics for open benchmarking, we recommend to use Recall@K, HitRate@K, and NDCG@K for candidate item matching tasks, and for CTR prediction tasks, we recommend to use logloss and AUC. We defer their definitions in later sections. These metrics are mostly used in the literature and usually adopted in practice. Our benchmarking work reports the results of these standard metrics to allow for easy reuse.

7) Benchmarking artifacts: Last but most importantly, our benchmarking pipeline produces the following seven artifacts as illustrated in Figure 2, which are necessary for benchmark reproducibility, including open datasets, dataset ID recording the specific data splitting, the evaluation protocol setting along with the dataset ID, open-source models, hyper-parameter configurations for model training, the running log and evaluation results for each experiment. Lastly, we document the detailed reproducing steps in a friendly format along with these artifacts, in order to ensure that all results are easily reproducible.

We note that, to the best of our knowledge, this is the first work that aims to build a standard and rigorous pipeline for reproducible research on recommender systems. This imposes new requirements on how researchers deliver their research results and how readers (or reviewers) assess the reproducibility of published papers. Missing any part of the above components defined in the open benchmarking pipeline could largely result in incomplete or broken research artifacts. Unfortunately, most of the existing work under evaluation fails to meet the criteria. We hope that the definition of our open benchmarking pipeline could enhance rigor in current research and offer good guidance for reproducible work.

4 BENCHMARKING FOR CANDIDATE ITEM MATCHING

4.1 Benchmarking Settings

Datasets: The datasets used for the research of candidate item matching are abundant. A lot of work chooses to perform evaluation on their private dataset splits and may use different experimental settings. This makes it difficult to directly make comparisons and accurately measure their progress. In this section, we use three representative public datasets, including AmazonBooks [1], Yelp2018 [6], and Gowalla [4], to present our preliminary benchmarking results. These three datasets have millions of interactions and are widely adopted to evaluate recommendation models, especially the state-of-the-art GNN-based models (e.g., NGCF [107], LightGCN [40]). For easy reuse in future research, we uniquely mark the three dataset splits as `AmazonBooks_m1`, `Yelp2018_m1`, and `Gowalla_m1`, respectively. We summarize the statistics of datasets in Table 2.

Evaluation Metrics: We adopt two commonly-used metrics, Recall@k and NDCG@k, in our evaluation. Especially, we set $k = 20$ as reported in [40].

- **Recall** [21]: Recall is the fraction of the user's interested items that are successfully retrieved. Higher recall indicates better retrieval performance for the recommender system.
- **NDCG** [109]: NDCG is a measure of ranking quality. Higher NDCG indicates that more items of users' interests are ranked at top by the recommender system.

Table 2: Datasets statistics for candidate item matching.

Dataset	#Users	#Items	#Interactions	Density
AmazonBooks	52, 643	91, 599	2, 984, 108	0.062%
Yelp2018	31, 668	38, 048	1, 561, 406	0.130%
Gowalla	29, 858	40, 981	1, 027, 370	0.084%

Note that all the evaluations are performed on the entire item corpus (i.e., no sampled metrics are used as suggested in [49]). This also conforms to the practical setting in industry where we usually resort to approximate nearest neighbor search tools (e.g., Faiss [45]) for efficient top-K retrieval even when the entire item corpus is extremely large. The full benchmarking results including more datasets and more metrics are available on the website.

Benchmarked Models: We comprehensively evaluate 33 popular models for candidate item matching in our benchmark, including 9 collaborative filtering-based models, 4 autoencoder-based models, 3 two-tower based models, 4 graph embedding-based models, and 13 GNN-based models. See Table 3 for the models and their references.

4.2 Benchmarking Results and Analysis

A systematic performance comparison is shown in Table 3. We show the results of different categories of models in different blocks for clear presentation. The top-5 results under each metric are shown in bold. We obtain several instructive findings from these clear benchmark results.

First, GNN-based item matching models are extremely popular in recent years. Performance progress has been gradually made in GNN-based models. From GC-MC, to NGCF, to LightGCN, and to the follow-up models, recommendation has become more and more accurate given more appropriate modeling. Meanwhile, GNN-based models have outperformed many traditional CF models (e.g., NeuMF), which to a certain extent, verifies the potential effectiveness of GNN-based models.

Second, except the graph embedding-based models, the best models in the other four categories are generally comparable. For example, with respect to Recall@20, the autoencoder-based model SLIM achieves the best performance on AmazonBooks while the two-tower based model SimpleX performs best on Yelp2018. Each category has models that perform within top-5, demonstrating that the item matching task has not been dominated by any types of methods and still has large development space. New breakthroughs are possible in any category. Furthermore, with the help of this clear benchmark results, researchers may more conveniently identify the bottlenecks and potential improvements of their own models through carefully considering the characteristics of the other models. For example, two well-performed models, i.e., ItemKNN and UltraGCN, leverage the item-item similarity graph, which may be crucial for performance improvement.

Last but most importantly, while many models are proposed with claims of achieving a state-of-the-art performance, they may fail to make a rigorous comparison to some extent. For example, ItemKNN and SLIM, which are two old models that proposed in 2001 and 2011 respectively, can outperform all of the other models

Table 3: Benchmarking results of existing models on candidate item matching. We highlight the top-5 best results in each column to indicate the SOTA performance. Note that some rows marked with asterisks are replicated from the existing papers (see the sources from the website leaderboard), but all the results are consistently reported under the same evaluation settings.

Year	Model	AmazonBooks_m1		Yelp2018_m1		Gowalla_m1	
		Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
CF-based Models							
2001	ItemKNN [85]	**0.0736(2)**	**0.0606(1)**	0.0639	0.0531	0.1570	0.1214
2012	MF-BPR [79]	0.0250	0.0196	0.0433	0.0354	0.1291	0.1109
2017	GRMF* [65]	0.0354	0.0270	0.0571	0.0462	0.1477	0.1205
2017	NeuMF* [41]	0.0258	0.0200	0.0451	0.0363	0.1399	0.1212
2017	CML [42]	0.0522	0.0428	0.0622	0.0536	0.1670	0.1292
2018	CMN* [34]	0.0267	0.0218	0.0457	0.0369	0.1405	0.1221
2018	HOP-Rec* [118]	0.0309	0.0232	0.0517	0.0428	0.1399	0.1214
2020	ENMF [23]	0.0359	0.0281	0.0624	0.0515	0.1523	0.1315
2021	MF-CCL [66]	0.0559	0.0447	**0.0698(2)**	**0.0572(2)**	**0.1837(5)**	0.1493
Autoencoder-based Models							
2011	SLIM [69]	**0.0755(1)**	**0.0602(2)**	0.0646	0.0541	0.1699	0.1382
2018	MultVAE* [56]	0.0407	0.0315	0.0584	0.0450	0.1641	0.1335
2019	MacridVAE* [63]	0.0383	0.0295	0.0612	0.0495	0.1618	0.1202
2019	EASE^R [93]	**0.0710(3)**	**0.0567(4)**	0.0657	0.0552	0.1765	0.1467
Two-Tower Models							
2016	YoutubeDNN [27]	0.0502	0.0388	**0.0686(4)**	**0.0567(4)**	0.1754	0.1473
2021	SimpleX [66]	0.0583	0.0468	**0.0701(1)**	**0.0575(1)**	**0.1872(1)**	**0.1557(3)**
Graph Embedding Models							
2014	DeepWalk [72]	0.0346	0.0264	0.0476	0.0378	0.1034	0.0740
2015	LINE [99]	0.0410	0.0318	0.0549	0.0446	0.1335	0.1056
2016	Node2Vec [36]	0.0402	0.0309	0.0452	0.0360	0.1019	0.0709
2016	Item2Vec [18]	0.0326	0.0251	0.0503	0.0411	0.1325	0.1057
GNN-based Models							
2017	GC-MC* [19]	0.0288	0.0224	0.0462	0.0379	0.1395	0.1204
2018	PinSage* [121]	0.0282	0.0219	0.0471	0.0393	0.1380	0.1196
2018	GAT* [101]	0.0326	0.0235	0.0543	0.0431	0.1501	0.1233
2019	NGCF* [107]	0.0344	0.0263	0.0579	0.0477	0.1570	0.1327
2019	DisenGCN* [61]	0.0329	0.0254	0.0558	0.0454	0.1356	0.1174
2020	LR-GCCF [24]	0.0335	0.0265	0.0561	0.0343	0.1519	0.1285
2020	NIA-GCN* [96]	0.0369	0.0287	0.0599	0.0491	0.1359	0.1106
2020	LightGCN* [40]	0.0411	0.0315	0.0649	0.0530	0.1830	**0.1554(4)**
2020	DGCF* [108]	0.0422	0.0324	0.0654	0.0534	**0.1842(4)**	**0.1561(2)**
2020	NGAT4Rec* [91]	0.0457	0.0358	0.0675	0.0554	–	–
2021	SGL-ED* [112]	0.0478	0.0379	0.0675	0.0555	–	–
2021	GF-CF* [88]	**0.0710(3)**	**0.0584(3)**	**0.0697(3)**	**0.0571(3)**	**0.1849(3)**	**0.1518(5)**
2021	UltraGCN [67]	**0.0681(5)**	**0.0556(5)**	**0.0683(5)**	**0.0561(5)**	**0.1862(2)**	**0.1580(1)**

Table 4: Statistics of the CTR prediction datasets.

Dataset	#Instances	#Fields	#Features	%Positives
Criteo	46M	39	5.55M	26%
Avazu	40M	24	8.37M	17%
KKBox	6.5M	13	0.1M	50.4%

on AmazonBooks. Similarly, YoutubeDNN [27], which is a well-known recommendation model proposed in 2016, surprisingly beats lots of newer models (e.g., HOP-Rec [118], MacridVAE [63], and NIA-GCN [96]) on the other two benchmark datasets. Besides, traditional graph embedding methods, such as DeepWalk, LINE, and Node2Vec, have been less compared, but they have been commonly used in industry as well (e.g., [103]). In our evaluation, we can see that early GNN-based models, e.g., NGCF, only have comparable

(or slightly worse) performance over them. In summary, some important baselines are unfortunately omitted by much recent work, resulting in more or less exaggeration of their improvements.

Such an undesirable phenomenon is largely due to the absence of a unified benchmark. In this case, if the early work accidentally misses some important comparisons, subsequent work based on it may easily repeat this omission that leads to a bad circle. We envision that our open benchmark will effectively improve this problem and inspire more solid and reproducible work to the community.

5 BENCHMARKING FOR CTR PREDICTION

5.1 Benchmarking Settings

Datasets: CTR prediction has been widely studied in the literature, where models have been evaluated on many different datasets and settings. In this section, we present our preliminary benchmarking results on three widely-used open datasets, including Criteo [3], Avazu [2], and KKBox [5]. We summarize the statistics of the datasets in Table 4. In contrast to existing studies that often take their private data splitting and preprocessing, we follow the work [92] for random train/validation/test splitting and release the prepossessing code to reduce the bias for benchmarking. The corresponding data splits are uniquely marked as `Criteo_x4`, `Avazu_x4`, and `KKBox_x1`, respectively. We refer readers to our website for more details about the benchmarking settings and the complete benchmarking results on more datasets.

Evaluation Metrics: As with many existing studies, we report the results on two most commonly used metrics, i.e., logloss and AUC.

1) **Logloss**: Logloss is also known as the binary cross-entropy loss, which is used to measure the loss for a binary classification problem. Lower logloss indicates better CTR prediction performance.

2) **AUC**: AUC [59] is commonly used to measure the probability that a randomly chosen positive sample is ranked higher than a randomly chosen negative sample. Higher AUC means better CTR prediction performance.

Benchmarked Models: We have comprehensively evaluated a total of 32 models for CTR prediction, which focus primarily on modeling feature interactions. Please refer to Table 5 for the list of models and their references.

5.2 Benchmarking Results and Analysis

Table 5 presents our benchmarking results on three datasets. We also highlight top-5 best results in bold face in each column. We have the following insightful observations from the results.

First, during the last decade, we have witnessed the evolution of CTR prediction models from simple logistic regression (LR), factorization machines (FM), to various deep models. We can observe that significant performance improvements have been made along this way. Successful deep models include DNN [27], Wide&Deep [25], DeepFM [37], DCN [105], etc., which have been widely adopted in industry.

Second, we can see that the top-5 best performing models differ widely on different datasets. That is, no one model can rule all the datasets, contradicted with the results reported by existing papers. Nevertheless, we can observe some robustly-performing models

including DeepFM [37], DCN [105], xDeepFM [55], ONN [120], and DCN-V2 [106]. We strongly suggest practitioners to try these robust models first for their practical problems.

Third, we note that recent performance gains of deep models, since the presence of DNN [27] and Wide&Deep [25], have become diminished. In many cases, different deep models only make a subtle difference (less than 0.1%) after sufficient model tuning. We note that the observation is consistent with the experimental results reported in some third-party papers[3] [106, 126], which validates its correctness. From another perspective, it also reveals that making strong improvements in large-scale CTR prediction tasks is difficult. Therefore, many recent studies tend to improve CTR prediction models from other potential aspects, such as behaviour sequence modeling [35, 127], multi-task modeling [64], and cross-domain modeling [70, 90].

Lastly, we intend to highlight that our benchmarking shows inconsistent results with those reported by some existing papers, which however claim large improvements therein. For example, DeepFM, DCN, xDeepFM, and DCN_v2 all achieve the same level of accuracy (around 81.4 AUC) on Criteo_x4, while PNN, HFM+, and FGCNN attain almost the same performance (around 79.44 AUC) on Avazu_x4. We run many experiments with different hyper-parameters and seeds, but do not obtain sufficiently distinct results. Especially, some recently proposed models, such as InterHAt [53], AFN+ [26], and LorentzFM [115], obtain even worse results than some previous baselines. This reveals the depressing fact that the performance improvements claimed by some recent papers are somewhat exaggerated.

Our results raise questions about rigor and reproducibility of the experimental results of existing work, and suggest that baseline models should be more carefully tuned to make more fair comparisons and promote healthy improvements. We hope that the availability of our BARS project could serve as a good starting point and facilitate more reproducible and convincing results in future research.

6 CALL FOR CONTRIBUTIONS

Openness is key to fostering progress in science via transparency and availability of all research outputs [12]. The evolution of open benchmarking is not possible without active support from the community. Towards this goal, we call for contributions from interested researchers, students, and practitioners. Specifically, everyone can contribute from either one of the following aspects: 1) expanding new datasets, 2) making dataset splits repeatable and shareable, 3) adding new models or new results, 4) verifying the reproducibility of existing models, 5) tuning existing models for better performance, 6) providing valuable feedbacks and suggestions, 7) last but most importantly, following the open benchmarking pipeline in future research. Every contribution will be appreciated and the contributor will be honored via a remark on the BARS website.

Future directions: While the current benchmark has covered both matching and ranking phases of recommender systems, there

---

[3]As reported in [106], DNN, PNN, DeepFM, xDeepFM, AutoInt+, and DCN have small AUC differences ($< 3e-4$) on Criteo. As reported in [126], Wide&Deep, DeepFM, DCN, and DCN-V2 have similar AUC (differences $< 4e-4$) on both Criteo and Avazu.

Table 5: Benchmarking results of different CTR prediction models. We highlight the top-5 best results in each column. Note that some of the results on Criteo_x4 and Avazu_x4 have been reported in our previous work [130]. Yet, we add substantially more results of new models and new datasets in this paper (See more results on the website leaderboard due to space limit.).

Year	Model	Criteo_x4		Avazu_x4		KKBox_x1	
		Logloss ($\times 10^{-2}$)	AUC(%)	Logloss ($\times 10^{-2}$)	AUC(%)	Logloss ($\times 10^{-2}$)	AUC(%)
2007	LR [82]	45.68	79.34	38.15	77.75	57.46	76.78
2010	FM [78]	44.31	80.86	37.54	78.87	50.75	82.91
2015	CCPM [58]	44.15	81.04	37.45	78.92	50.13	83.72
2016	FFM [46]	44.07	81.13	37.20	79.31	49.74	83.76
2016	HOFM [20]	44.11	81.07	37.54	78.91	50.48	83.15
2016	PNN [76]	43.78	81.42	**37.12(4)**	**79.44(3)**	47.93	85.15
2016	DNN [27]	43.80	81.40	37.22	79.28	48.11	85.01
2016	Wide&Deep [25]	43.77	81.42	37.20	79.29	48.52	85.04
2016	DeepCrossing [86]	43.84	81.35	37.21	79.30	47.99	84.95
2017	NFM [39]	44.24	80.93	37.43	78.94	51.02	82.85
2017	AFM [114]	44.55	80.60	37.93	78.23	52.41	81.75
2017	DeepFM [37]	**43.76(3)**	**81.43(5)**	37.19	79.30	47.85	**85.31(4)**
2017	CrossNet [105]	44.56	80.60	37.79	78.40	52.83	81.16
2017	DCN [105]	**43.76(3)**	**81.44(4)**	37.19	79.31	**47.66(1)**	**85.31(4)**
2018	FwFM [71]	44.08	81.12	37.44	79.07	49.71	84.06
2018	CIN [55]	43.94	81.27	37.42	78.94	49.09	84.26
2018	xDeepFM [55]	**43.76(3)**	**81.43(5)**	37.18	79.33	**47.72(2)**	**85.35(2)**
2019	FiGNN [54]	43.83	81.38	37.36	79.15	48.96	84.72
2019	FiBiNET [43]	43.87	81.31	**37.05(2)**	**79.53(2)**	48.14	84.99
2019	AutoInt [92]	43.99	81.19	37.45	78.91	49.19	84.36
2019	AutoInt+ [92]	43.90	81.32	37.46	79.02	**47.73(3)**	**85.34(3)**
2019	HFM [100]	44.24	80.95	37.57	78.79	49.70	83.92
2019	HFM+ [100]	43.92	81.27	37.14	**79.44(3)**	**47.81(5)**	85.21
2019	FGCNN [57]	43.98	81.21	**37.11(3)**	**79.44(3)**	48.01	85.22
2020	LorentzFM [115]	44.34	80.83	37.56	78.85	51.88	82.02
2020	InterHAt [53]	44.14	81.04	37.49	78.82	48.63	84.59
2020	AFN [26]	44.02	81.15	37.40	79.07	49.10	84.26
2020	AFN+ [26]	43.84	81.38	37.26	79.29	48.42	84.89
2020	DeepIM [122]	**43.75(2)**	**81.46(2)**	**37.16(5)**	79.35	**47.75(4)**	**85.37(1)**
2020	ONN [120]	**43.72(1)**	**81.48(1)**	**36.83(1)**	**79.92(1)**	48.56	84.98
2021	FmFM [97]	43.97	81.24	37.47	79.01	49.46	84.07
2021	DCN-V2 [106]	**43.75(2)**	**81.45(3)**	37.19	79.31	47.87	**85.31(4)**

are still many other tasks that could be incorporated in future versions, such as re-ranking and sequential recommendation. In addition, we could establish specialised benchmarking leaderboards for vertical recommendation scenarios, such as news recommendation, music recommendation, and micro-video recommendation. Moreover, current benchmarking results are only accuracy-focused, it is desirable to add more evaluation metrics such as diversity, coverage, fairness, and so on. Building such a comprehensive open benchmark is a challenging yet worthwhile task for the recommender systems community. We call for active support and contributions in any form from the community to make it possible.

7 CONCLUSION

Reproducibility is an open issue in the field of recommender systems. In this paper, we aim to build an open benchmark for recommender systems, covering both candidate item matching and CTR prediction tasks, to drive more solid and reproducible research in this field. Towards this end, we set up a standardized benchmarking pipeline, establish a benchmark website, and deliver the most comprehensive benchmarking results to date. We hope that the project could benefit all researchers, practitioners, and educators in the community.

---

[4]https://www.mindspore.cn

REFERENCES

[1] 2021. The AmazonBooks Dataset. https://jmcauley.ucsd.edu/data/amazon/amazonbooks

[2] 2021. The Avazu Dataset. https://www.kaggle.com/c/avazu-ctr-prediction/data

[3] 2021. The Criteo Dataset. https://www.kaggle.com/c/criteo-display-ad-challenge/data

[4] 2021. The Gowalla Dataset. https://snap.stanford.edu/data/loc-gowalla.html

[5] 2021. The KKBox Dataset. https://www.kaggle.com/c/kkbox-music-recommendation-challenge

[6] 2021. The Yelp Dataset. https://www.yelp.com/dataset

[7] 2022. DeepCTR: Easy-to-use, modular and extendible package of deep-learning based CTR models. https://github.com/shenweichen/deepctr

[8] 2022. EasyRec: A tensorflow framework for large scale recommendation algorithms. https://github.com/alibaba/EasyRec

[9] 2022. FuxiCTR: A configurable, tunable, and reproducible library for CTR prediction. https://github.com/xue-pai/FuxiCTR

[10] 2022. Milvus: Vector database built for scalable similarity search. https://milvus.io

[11] 2022. The MovieLens Dataset. https://grouplens.org/datasets/movielens

[12] 2022. Open Science Policies. https://conf.researchr.org/track/icse-2022/icse-2022-open-science-policies

[13] 2022. PaddleRec: A library of recommendation models based on PaddlePaddle. https://github.com/PaddlePaddle/PaddleRec

[14] 2022. TensorFlow Recommenders: A library for building recommender system models using TensorFlow. https://github.com/tensorflow/recommenders

[15] 2022. TorchRec: A Pytorch domain library for recommendation systems. https://github.com/pytorch/torchrec

[16] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, and Claudio Pomo. 2021. Reenvisioning the comparison between Neural Collaborative Filtering and Matrix Factorization. In *Proceedings of the 15th ACM Conference on Recommender Systems (RecSys)*. 521–529.

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

[18] 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)*. 1–6.

[19] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. In *KDD'18 Deep Learning Day*.

[20] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-Order Factorization Machines. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 3351–3359.

[21] Michael Buckland and Fredric Gey. 1994. The relationship between recall and precision. *Journal of the American society for information science* 45, 1 (1994), 12–19.

[22] Guohao Cai, Jieming Zhu, Quanyu Dai, Zhenhua Dong, Xiuqiang He, Ruiming Tang, and Rui Zhang. 2022. ReLoop: A Self-Correction Continual Learning Loop for Recommender Systems. In *The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[23] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–28.

[24] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 27–34.

[25] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, et al. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS@RecSys)*. 7–10.

[26] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 3609–3616.

[27] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM conference on Recommender Systems (RecSys)*. 191–198.

[28] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys)*. 101–109.

[29] Maurizio Ferrari Dacrema, Federico Parroni, Paolo Cremonesi, and Dietmar Jannach. 2020. Critically Examining the Claimed Value of Convolutions over User-Item Embedding Maps for Recommender Systems. In *The ACM International Conference on Information and Knowledge Management (CIKM)*. 355–363.

[30] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2018. Adversarial Network Embedding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*. 2167–2174.

[31] Quanyu Dai, Xiao Shen, Liang Zhang, Qiang Li, and Dan Wang. 2019. Adversarial Training Methods for Network Embedding. In *The World Wide Web Conference (WWW)*. 329–339.

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

[33] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*. 2230–2236.

[34] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR)*. 515–524.

[35] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*. 2301–2307.

[36] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 855–864.

[37] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 1725–1731.

[38] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Vol. 119. 3887–3896.

[39] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 355–364.

[40] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong 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 (SIGIR)*. 639–648.

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

[42] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*. 193–201.

[43] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: Combining Feature Importance and Bilinear Feature Interaction for Click-Through Rate Prediction. In *Proceedings of ACM Conference on Recommender Systems (RecSys)*. 169–177.

[44] Dietmar Jannach, Oren Sar Shalom, and Joseph A. Konstan. 2019. Towards More Impactful Recommender Systems Research. In *Proceedings of the 1st Workshop on the Impact of Recommender Systems*.

[45] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-Scale Similarity Search with GPUs. *arXiv:1702.08734* (2017).

[46] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. 43–50.

[47] Joseph A. Konstan and Gediminas Adomavicius. 2013. Toward Identification and Adoption of Best Practices in Algorithmic Recommender Systems Research. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys)*. 23–28.

[48] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

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

[50] Adit Krishnan, Mahashweta Das, Mangesh Bendre, Hao Yang, and Hari Sundaram. 2020. Transfer Learning via Contextual Invariants for One-to-Many Cross-Domain Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1081–1090.

[51] Justin J. Levandoski, Michael D. Ekstrand, Michael Ludwig, Ahmed Eldawy, Mohamed F. Mokbel, and John Riedl. 2011. RecBench: Benchmarks for Evaluating Performance of Recommender System Architectures. *Proceedings of VLDB Endow.* 4, 11 (2011), 911–920.

[52] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. 2019. Label Efficient Semi-Supervised Learning via Graph Filtering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 9582–9591.

[53] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable Click-Through Rate Prediction through Hierarchical Attention. In *The International Conference on Web Search and Data Mining (WSDM)*. 313–321.

[54] 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 (CIKM).* 539–548.

[55] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, et al. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (KDD).* 1754–1763.

[56] 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.

[57] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In *The World Wide Web Conference, (WWW).* 1119–1129.

[58] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A Convolutional Click Prediction Model. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM).* 1743–1746.

[59] Jorge M Lobo, Alberto Jiménez-Valverde, and Raimundo Real. 2008. AUC: a Misleading Measure of the Performance of Predictive Distribution Models. *Global ecology and Biogeography* 17, 2 (2008), 145–151.

[60] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Performance Comparison of Neural and Non-neural Approaches to Session-based Recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys).* 462–466.

[61] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled Graph Convolutional Networks. In *International Conference on Machine Learning (ICML).* 4212–4221.

[62] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD).* 1930–1939.

[63] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning Disentangled Representations for Recommendation. In *Advances in Neural Information Processing Systems (NeurIPS).* 5711–5722.

[64] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *Proceedings of the 41st International Conference on Research & Development in Information Retrieval (SIGIR).* 1137–1140.

[65] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2017. A Deep Recurrent Collaborative Filtering Framework for Venue Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM).* 1429–1438.

[66] 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 2021 ACM on Conference on Information and Knowledge Management (CIKM).* 1243–1252.

[67] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In *Proceedings of the 2021 ACM on Conference on Information and Knowledge Management (CIKM).* 1253–1262.

[68] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad Click Prediction: a View from the Trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).* 1222–1230.

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

[70] Wentao Ouyang, Xiuwu Zhang, Lei Zhao, Jinmei Luo, Yu Zhang, Heng Zou, Zhaojie Liu, and Yanlong Du. 2020. MiNet: Mixed Interest Network for Cross-Domain Click-Through Rate Prediction. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM).* 2669–2676.

[71] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted Factorization Machines for Click-Through Rate Prediction in Display Advertising. In *Proceedings of the World Wide Web Conference on World Wide Web (WWW).* 1349–1357.

[72] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).* 701–710.

[73] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD).* ACM, 2671–2679.

[74] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM).* ACM, 2685–2692.

[75] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User Behavior Retrieval for Click-Through Rate Prediction. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (SIGIR).* 2347–2356.

[76] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-Based Neural Networks for User Response Prediction. In *Proceedings of the IEEE 16th International Conference on Data Mining (ICDM).* 1149–1154.

[77] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, and Kun Gai. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR).* 565–574.

[78] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM).* 995–1000.

[79] 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 (UAI).* 452–461.

[80] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *Fourteenth ACM Conference on Recommender Systems.* 240–248.

[81] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web (WWW).* 521–530.

[82] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting Clicks: Estimating the Click-Through Rate for New Ads. In *Proceedings of the 16th International Conference on World Wide Web, (WWW).* 521–530.

[83] Noveen Sachdeva and Julian J. McAuley. 2020. How Useful are Reviews for Recommendation? A Critical Review and Potential Improvements. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (SIGIR).* 1845–1848.

[84] Alan Said and Alejandro Bellogín. 2014. Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys).* 129–136.

[85] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of International Conference on World Wide Web (WWW).* 285–295.

[86] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. Mao. 2016. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).* 255–262.

[87] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *Proceedings of the 23rd international conference on world wide web (WWW).* 373–374.

[88] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, Khaled B Letaief, and Dongsheng Li. 2021. How Powerful is Graph Convolution for Recommendation?. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM).* 1619–1629.

[89] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I. Nikolenko. 2020. RecVAE: A New Variational Autoencoder for Top-N Recommendations with Implicit Feedback. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM).* 528–536.

[90] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, and Xiaoqiang Zhu. 2021. One Model to Serve All: Star Topology Adaptive Recommender for Multi-Domain CTR Prediction. In *The ACM International Conference on Information and Knowledge Management (CIKM).* 4104–4113.

[91] Jinbo Song, Chao Chang, Fei Sun, Xinbo Song, and Peng Jiang. 2020. NGAT4Rec: Neighbor-Aware Graph Attention Network For Recommendation. *arXiv preprint arXiv:2010.12256* (2020).

[92] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM).* 1161–1170.

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

[94] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Adv. Artif. Intell.* 2009 (2009), 421425:1–421425:19.

[95] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM).* 1441–1450.

[96] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. 2020. Neighbor Interaction Aware Graph Convolution Networks for Recommendation. In *Proceedings of the 43rd International*

ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). 1289–1298.

[97] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-matrixed Factorization Machines for Recommender Systems. In *Proceedings of the Web Conference 2021 (WWW)*. 2828–2837.

[98] 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 *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys)*. 23–32.

[99] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*. 1067–1077.

[100] Yi Tay, Shuai Zhang, Anh Tuan Luu, Siu Cheung Hui, Lina Yao, and Tran Dang Quang Vinh. 2019. Holographic Factorization Machines for Recommendation. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*. 5143–5150.

[101] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

[102] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *7th International Conference on Learning Representations (ICLR)*.

[103] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 839–848.

[104] Jinpeng Wang, Jieming Zhu, and Xiuqiang He. 2021. Cross-Batch Negative Sampling for Training Two-Tower Recommenders. In *The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1632–1636.

[105] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the 11th International Workshop on Data Mining for Online Advertising (ADKDD)*. 12:1–12:7.

[106] 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 (WWW)*. 1785–1797.

[107] 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 (SIGIR)*. 165–174.

[108] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1001–1010.

[109] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. 2013. A Theoretical Analysis of NDCG Ranking Measures. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT)*, Vol. 8. 6.

[110] Yikai Wang, Liang Zhang, Quanyu Dai, Fuchun Sun, Bo Zhang, Yang He, Weipeng Yan, and Yongjun Bao. 2019. Regularized Adversarial Sampling and Deep Time-aware Attention for Click-Through Rate Prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 349–358.

[111] Bin Wu, Zhongchuan Sun, Xiangnan He, Xiang Wang, and Jonathan Staniforth. 2020. NeuRec: An Open Source Neural Recommender Library. https://github.com/wubinzzu/NeuRec.

[112] 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 (SIGIR)*. 726–735.

[113] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 153–162.

[114] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI)*. 3119–3125.

[115] Canran Xu and Ming Wu. 2020. Learning Feature Interactions with Lorentzian Factorization Machine. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 6470–6477.

[116] Jiahao Xun, Shengyu Zhang, Zhou Zhao, Jieming Zhu, Qi Zhang, Jingjie Li, Xiuqiang He, Xiaofei He, Tat-Seng Chua, and Fei Wu. 2021. Why Do We Click: Visual Impression-aware News Recommendation. In *ACM Multimedia Conference (MM)*. 3881–3890.

[117] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations. In *Companion Proceedings of the Web Conference (WWW)*. 441–447.

[118] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-Rec: High-order Proximity for Implicit Recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys)*. 140–144.

[119] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. Critically Examining the "Neural Hype": Weak Baselines and the Additivity of Effectiveness Gains from Neural Ranking Models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1129–1132.

[120] Yi Yang, Baile Xu, Shaofeng Shen, Furao Shen, and Jian Zhao. 2020. Operation-aware Neural Networks for User Response Prediction. *Neural Networks* 121 (2020), 161–168.

[121] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 974–983.

[122] Feng Yu, Zhaocheng Liu, Qiang Liu, Haoli Zhang, Shu Wu, and Liang Wang. 2020. Deep Interaction Machine: A Simple but Effective Model for High-order Feature Interactions. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM)*. 2285–2288.

[123] 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)*.

[124] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. 2021. Deep Learning for Click-Through Rate Estimation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*. 4695–4703.

[125] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, et al. 2021. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM)*. 4653–4664.

[126] Zangwei Zheng, Pengtai Xu, Xuan Zou, Da Tang, Zhen Li, Chenguang Xi, Peng Wu, Leqi Zou, Yijie Zhu, Ming Chen, Xiangzhuo Ding, Fuzhao Xue, Ziheng Qing, Youlong Cheng, and Yang You. 2022. CowClip: Reducing CTR Prediction Model Training Time from 12 Hours to 10 Minutes on 1 GPU. *CoRR* abs/2204.06240 (2022).

[127] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*. 5941–5948.

[128] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1059–1068.

[129] Jieming Zhu, Jinyang Liu, Weiqi Li, Jincai Lai, Xiuqiang He, Liang Chen, and Zibin Zheng. 2020. Ensembled CTR Prediction via Knowledge Distillation. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM)*. 2941–2958.

[130] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open Benchmarking for Click-Through Rate Prediction. In *The 30th ACM International Conference on Information and Knowledge Management (CIKM)*. 2759–2769.