Empirically Testing Deep and Shallow Ranking Models for Click-Through Rate (CTR) prediction

Yi-Che Yang, Ping-Ching Lai, Hung-Hsuan Chen
Computer Science and Information Engineering
National Central University
Taoyuan, Taiwan
yiche@g.ncu.edu.tw, E19960102@gmail.com, hhchen@g.ncu.edu.tw

***Abstract*—Recent studies have reported that deep learning models perform excellently for reranking the top recommendation items. However, we found that it is not easy to reproduce some of these results. In particular, we found that recommendations based on a simple neighbor-based model, on average, outperform the results generated by deep learning models based on two datasets from e-commerce websites (one open dataset and one private dataset from our collaborating partner). Moreover, we performed an error analysis to investigate when the deep learning models perform better than simple models and when they do not. Our analysis is especially useful for medium- and small-sized online retailers that may have a smaller training dataset.**

***Index Terms*—recommender systems, ranking models, deep learning, match and rank, word embedding**

I. Introduction

As deep learning is successful and has become popular in many domains, it is not surprising that online retailers utilize this new tool to boost the performance of recommender systems. Many papers have indeed reported performance improvements when using deep learning for recommendation tasks, especially for the click-through rate (CTR) prediction tasks [1]–[5]. To facilitate research and reproducibility, many authors have released their experimental code, and some researchers have even implemented packages, e.g., DeepCTR,[1] to integrate various deep learning-based recommendation modules so that these models have a unified programming interface.

As we experimented with many of these models using our dataset collected from a medium-sized e-commerce company, we found that it is difficult to reproduce a similar degree of improvement that was reported in these papers. In certain cases, these sophisticated deep learning models performed even worse than simple models, such as recommending the k-nearest neighbors based on the cosine similarity between the object embeddings generated by self-supervised models, such as Prod2Vec [6] or Behavior2Vec [7]. These results lead us to consider the following questions. First, why do deep learning models usually fail on our dataset? Second, when and how should we apply sophisticated deep learning models for recommendations?

To answer the above questions, we conducted error analyses on deep learning recommendation models and the simple nearest-neighbor recommendation model. We found that deep learning models seem to work well when there are enough "clues" in the training data. We also found that the authors who have proposed deep learning recommendations models mostly work for Internet giants, e.g., Amazon [8], Alibaba (and its subgroup Taobao) [2], [4], [5], Sina Weibo [1], and Huawei [3]. These companies can easily access the logs that may include the behaviors of hundreds of millions of users. Consequently, training complex deep learning models is less likely to result in overfits, as the size of the training data is enormous. On the other hand, we surmise that for small- and medium-sized e-commerce websites, applying deep learning models to learn users' preferences may easily overfit the training data, so using simple models may end up obtaining better recommendation lists on average. That being said, we still show that, in certain cases, small- and medium-sized companies should apply the more sophisticated recommendation algorithms. In particular, if an item frequently appears in the training data, we may have more clues about this item, and the deep learning recommendation models will tend to make better recommendations for this item. As most e-commerce companies are small- or medium-sized,[2] our study may benefit the majority of the e-commerce companies.

The rest of the paper is organized as follows. In Section II, we review previous works on recommendation algorithms, especially the deep learning-based recommendation algorithms. Section III shows the experimental methods, settings, and datasets. Section IV reports the experimental results of the compared models. Finally, we discuss the discoveries and the limitations of our study and the ongoing and future works in Section V.

II. Related Work

Collaborative filtering algorithms utilize multiple users' collective behaviors to decide the recommended items for users, so no item description or user profiles are needed. Early recommendation models on collaborative filtering mostly leverage linear models or their variations [9]–[13]. Recent studies have

We acknowledge partial support by the Ministry of Science and Technology under grant no. MOST 107-2221-E-008-077-MY3.

[1]https://github.com/shenweichen/DeepCTR

[2]In 2019, PipeCandy estimated more than 2 million e-commerce companies in the world, excluding companies in China. https://blog.pipecandy.com/e-commerce-companies-market-size/

mainly adapted deep learning models for recommendation and hoped that these sophisticated models could discover the high-dimensional nonlinear patterns among the raw features and the targets. This line of research includes Wide&Deep [14], DeepFM [15], NeuralCF [16], to name a few. Essentially, these models generate the latent representations of the users and items and further utilize the interaction of these latent representations to make recommendations. However, these methods usually require a fixed number of features. If we use a user's previous behaviors (e.g., a series of clicks) as the input features, the number of features may vary (as the lengths of different users' clickstreams are different), so applying the above methods may require preprocessing (e.g., by average pooling), which may inevitably lose certain information.

A natural alternative to solve the above issue is to adopt a recurrent neural network (RNN) or one of its variants and treat the behavior sequence as the model input [17], [18]. However, the RNN and RNN variants are usually challenging to train. Moreover, older (earlier) information may gradually decay. More advanced structures (e.g., LSTM and GRU) can only slow the decay process, but eventually, earlier information is still likely to be lost. Consequently, researchers have proposed applying the attention-based approach [19] so that the weights of each previous action are decided on-the-fly. However, attention-based models do not preserve the sequential information by its nature. Consequently, positional embeddings might be needed to retain the order of the clicked items [19].

Although recommender systems have been studied for decades, recent studies have started to re-examine the practical performance of these various recommendation algorithms and the limitations on the evaluation strategies. It was found that offline evaluation cannot fully represent the online performance of an algorithm because of offline evaluation bias [20], [21]. Even if the offline evaluation well represents the online performance of a recommendation algorithm, reproducing the experimental results on recent deep learning-based recommendation models is difficult [22]. Even worse, a recent study showed that many of the deep learning-based recommendation models could be outperformed by simple recommendation algorithms [22]. This result is consistent with our experiments. However, we further show when deep learning recommendation models may be useful and why they may or may not work.

III. Experimental Method

This section presents the current setting of recommender systems for most e-commerce companies, and how we compared different models based on the current setting.

A. Preliminaries: Candidate Generation and Ranking Mechanism

For an e-commerce website that contains millions of products that must be recommended efficiently, the recommendation workflow usually comprises two steps: candidate generation and ranking [23]. The candidate generation step generates a few (typically hundreds) of candidate items that are likely to interest a user. Since this step requires selection from a vast number of items, the underlying algorithm is usually simple and fast. Sometimes, these algorithms are performed in advance, and the results are stored for online queries. The ranking step fine-tunes the ranking of the selected candidates. The ranking algorithm can be sophisticated because of the number of the candidate items is small. Most recent studies on improving the click-through rate (CTR) focus on developing different types of ranking models based on deep learning [4], [5].

We followed this candidate generation and ranking mechanism for the following experiments. In the candidate generation step, we applied the Prod2Vec algorithm [6] to generate the embeddings of each item. Essentially, the Prod2Vec model treats each item as a word and each clickstream as a sentence so that we may leverage word embedding models [24] to generate the embedding of each item. When a user browses an item i (which we call an anchor item in this paper), the recommendation model selects 100 items whose embeddings are most similar to the embedding of item i (in terms of the cosine similarity) as the candidates. We investigated and compared the effectiveness of different ranking models to rerank the top 100 candidates.

B. Ranking Models

Once candidate items are generated, the ranking algorithm reranks these candidates to show the items the user is most likely to be interested in at the top of the recommendation list. Recent studies typically used sophisticated deep learning strategies (e.g., CNNs, RNNs, and attention mechanisms) for reranking because deep learning models can capture the high-dimensional, nonlinear relationships among different embeddings. This section introduces the ranking models we compared.

The first ranking model is a baseline model for comparison. This model simply ranks the candidates based on the cosine similarity between the embeddings of the anchor item and the candidate items. Essentially, this is simply the Prod2Vec model — the top candidates are selected without reranking them. We abbreviate this model `KNN` in the following, as this model ranks the k-nearest neighbors to the anchor item as the recommendation list.

The second model is a multilayer perceptron model, which is probably the most straightforward ranking model based on deep learning. The model takes the embedding e_i of the anchor item and the embedding e_j of a candidate item as the inputs and predicts the probability $f(e_i, e_j)$ that a user clicks item j after browsing item i using a multilayer perceptron. Specifically, during the training phase, if a user indeed clicks on item j after browsing item i, this pair is regarded as a positive instance (i.e., $f(e_i, e_j) = 1$). The negative instances are obtained via negative sampling [25]: let e_k be the embedding of the sampled negative instance, we want $f(e_i, e_k) = 0$. Note that we do not change the embeddings of the items here but merely try to optimize the parameters of the network. At the inference phase, the algorithm needs to compute $f(e_i, e_k)$ for every candidate item k and ranks the candidate items by their clicking probabilities. We abbreviate this model NN in the following, as it utilizes a neural network architecture. Note that the NN model here is very close to the BaseModel introduced in [4], [5]. Compared to the baseline KNN model, the new model allows high-dimensional interactions between the elements in the embeddings. Hence, it is likely to capture high-dimensional relationships among the embeddings. Many recent studies utilized this design as a base model and introduced different features or different network structures to construct various ranking functions (fs) [4], [5], [23], [26].

The third model is the DIN model [5], which incorporates personal embedding as part of the inputs. This model utilizes an attention mechanism to determine the relative importance of a user's browsed items and further generate each user's personal embeddings. The DIN model can be regarded as an extension of YouTube's personalized recommendation model [23]: YouTube integrated a user's personal embedding by averaging the embeddings of the user's watched videos, whereas DIN learns personal embedding using the attention mechanism.

The fourth model is the DIEN model [4], which is an extension of the DIN model by modifying the network architecture. The main difference is that the DIEN model incorporates both GRU and the attention mechanism. Consequently, a user's previous visits are constructed as a sequence. The authors reported an 11.8% CTR gain compared to the DIN model [4] from online A/B testings on the advertising system of Taobao, the most visited e-commerce website in 2020, according to Alexa.[3]

C. Experimental Datasets

We evaluated the above ranking models using one open dataset and one private dataset. Table I shows the statistical summary of the two datasets.

The first dataset — users' behaviors on Taobao — was released by the Alibaba Group on its official data-sharing website Tianchi.[4] We used this dataset because the DIN and DIEN models were applied to Taobao. However, the original papers of these two models did not grant access the experimental dataset. Under such a scenario, the Taobao dataset on Tianchi is the most similar dataset we could obtain. We used the logs of the first day as the training data and the remaining logs as the test data. We further divide the training data into training and validation sets if hyperparameter tuning is needed.

We obtained the second dataset from a medium-sized e-commerce company in Taiwan. We call this dataset Retailer-X since the company prefers to remain anonymous. As shown in Table I, even though the period of the Retailer-X dataset covers a much longer period than the Taobao dataset (122 days vs. 9 days, respectively), the Taobao dataset has much more sessions. However, as the size of most e-commerce websites are closer to that of Retailer-X, perhaps our experience and experimental results for Retailer-X are more helpful for most retailers than those of Taobao. We used the logs of the first 14 days as the training data and the remaining logs as the test data. If a model requires hyperparameter tuning, we further divide the training data into training and validation sets.

D. Evaluation

We selected all the sessions that have at least two item-browsing behaviors as the experimental data. Each selected session of length ℓ is divided into two parts: the first $\ell - 1$ page views are used as the clues, and we want to predict the last item that the user is going to visit in this session based on the first $\ell - 1$ page views. To measure and compare the effectiveness of different learning algorithms, each algorithm ranks the candidate items by their predicted probabilities to be viewed.

Because we are dealing with a ranking task, natural choices of evaluation measurements are various ranking metrics, such as the precision@k, discounted cumulative gain (DCG), and normalized discounted cumulative gain (NDCG) [28]. However, as a session can have only one last visited item, we cannot be sure of the quality of rankings between the unclicked candidate items, so the standard ranking metrics may not be the best option here. Finally, we evaluated the effectiveness of each algorithm in three different ways. First, we compared the average positions of the clicked item in the recommendation lists generated by the different ranking algorithms. If an algorithm tended to rank the clicked item near the top of the recommendation list, it is considered a decent algorithm. Second, we compared the winning rates of these ranking algorithms. We define a win for algorithm i if this algorithm ranks the clicked item at the highest position among the compared algorithms, and the winning rate is the percentage of wins out of all trials. Third, we reported average precision@k ($k = 5, 10, 15, 20$), which is defined as the proportion of the recommended items in the top k that are clicked. However, as each session has one last visited item, the numerator of precision@k here can only be 0 or 1.

IV. Experimental Results

This section reports the compared models based on (1) the average position of the next clicked item in the recommendation list, (2) average precision@k, and (3) winning rate. We also discuss when the deep learning models erform better than simple models and when they do not.

A. Overall Performance

Table II shows the average position of the clicked item for the different ranking algorithms. The simple KNN algorithm performs slightly better than the complicated NN model and much better than the even more complicated DIN and DIEN models on both datasets. If we consider the metric average precision@k with $k = 5, 10, 15, 20$, the results are similar, as shown in Table III and Table IV. However, the authors who

[3]https://www.alexa.com/siteinfo/taobao.com

[4]https://tianchi.aliyun.com/

	Period	# of sessions	# of distinct products
Taobao	2017/11/25 – 2017/12/03	6,761,250	4,067,842
Retailer-X	2019/3/1 – 2019/6/30	2,639,734	592,768

TABLE I
A STATISTICAL SUMMARY OF THE TWO EXPERIMENTAL DATASETS. THE STATISTICAL SUMMARY OF THE FIRST DATASET IS DIFFERENT FROM THAT REPORTED IN [27] BECAUSE WE CONSIDER ONLY ITEM-VIEWING BEHAVIORS; ALL OTHER BEHAVIORS (E.G., PUTTING AN ITEM INTO THE SHOPPING CART, ADDING AN ITEM TO A FAVORITES LIST, ETC.) ARE EXCLUDED.

	KNN [6]	NN [4], [5], [23]	DIN [5]	DIEN [4]
Taobao	30.08	32.00	47.66	50.02
Retailer-X	20.29	20.62	44.65	46.63

TABLE II
AVERAGE POSITION OF THE CLICKED ITEM FOR THE DIFFERENT RANKING ALGORITHMS (THE SMALLER THE VALUE IS, THE MORE ACCURATE THE MODEL)

	KNN	NN	DIN	DIEN
Taobao	40%	30%	15%	15%
Retailer-X	45%	34%	10%	11%

TABLE V
WINNING RATES OF THE COMPARED RANKING ALGORITHMS

	KNN [6]	NN [4], [5], [23]	DIN [5]	DIEN [4]
precision@5	24%	20%	5%	6%
precision@10	35%	30%	10%	10%
precision@15	43%	39%	15%	15%
precision@20	50%	46%	21%	20%

TABLE III
AVERAGE PRECISION@k FOR THE DIFFERENT RANKING ALGORITHMS ON THE TAOBAO DATASET

	KNN [6]	NN [4], [5], [23]	DIN [5]	DIEN [4]
precision@5	38%	33%	5%	8%
precision@10	52%	48%	12%	17%
precision@15	61%	58%	18%	24%
precision@20	67%	65%	25%	29%

TABLE IV
AVERAGE PRECISION@k FOR THE DIFFERENT RANKING ALGORITHMS ON THE RETAILER-X DATASET

proposed the DIN and DIEN models reported that the NN model performed worse than the DIN model and the DIEN model [4], [5]. That being said, there are differences between our experimental settings and the ones reported in [4], [5]. First, the authors of the DIN and DIEN models compared the ranking algorithms based on the area under the ROC curve. In contrast, we used a more straightforward metric — the position of the clicked item in the recommendation list. Second, we cannot determine how the authors who proposed the DIN and DIEN models generated the item embeddings. In our experiments, we first produced item embeddings by the Prod2Vec model, and then we fixed the item embeddings and simply trained the parameters in the neural network. We applied this procedure because it becomes a common process of conducting NLP projects: word embeddings are pretrained and then these embeddings are used as inputs of a neural network to train a particular task. If the authors of [4], [5] generated the item embeddings in other ways, e.g., training the item embeddings along with the network parameters for the NN model, then perhaps it is the embedding generating process that caused the difference between our NN model and those reported in [4], [5].

We also evaluated the different ranking models based on their winning rates. Particularly, a ranking algorithm r_i wins if it ranks the next clicked item highest among the four compared models, and the winning rate of r_i is simply the percentage of wins that r_i achieves.

Table V lists the winning rates of the four compared methods. It appears that KNN and NN usually rank the next clicked item better than the other two algorithms, but the DIN and DIEN models may still win occasionally (approximately 10% to 15% of the time). Although, on average, the simple models (e.g., KNN and perhaps NN) seem to outperform the others, a better strategy might be combining the four models, as there appears to be no single winner in all cases. Therefore, our question becomes the following. First, why did the earlier papers report that sophisticated models performed better on average? Second, when should we apply complicated ranking algorithms, and when should we use simple ranking algorithms?

B. Why Previous Papers Reported Mediocre Results on Simple Models

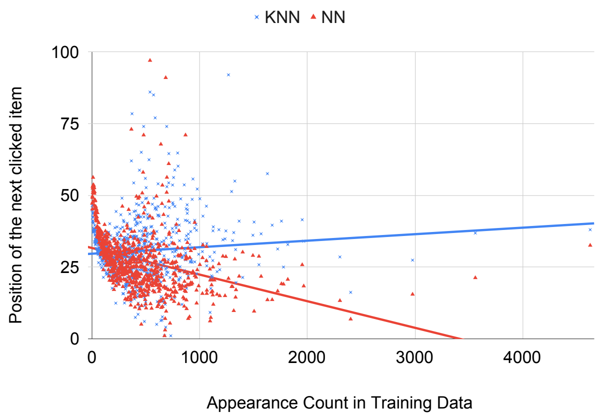
Bias and variance analysis is a useful tool to analyze why a learning algorithm fails [29], [30]. We suspect that the previous papers reported better results for the complex models because these models were trained on larger training data. To validate this hypothesis, we conducted the following two experiments. First, assuming that a user clicked on item j after viewing item i, we compared the relationship between the appearance count of i in the training data and the position of j in the recommendation list. We presume that if an item i appears frequently in the training data, the models may obtain more information about this item i and therefore generate a better recommendation list when the anchor item is item i. Such a phenomenon should be more evident for the complicated models, as they are more likely to overfit the training data. Second, we allocated a larger ratio of labeled instances to the training data and observed how the results varied.

Figure 1 shows the first result: the relationship between the number of appearances of an item in the training data and the position of the next clicked item in the recommendation list (smaller is better). We compared the KNN model and the NN model. As shown, when an item appears frequently in the training data, the NN model tends to rank the next clicked item closer to the top of the recommendation list (i.e., the

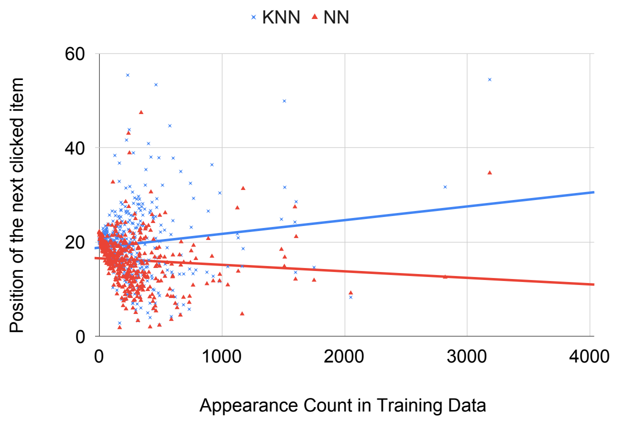
Training data size	Taobao 1 day	Taobao 4 days	Retailer-X 14 days	Retailer-X 2 months
Average rank of DIN	47.808	33.251	44.331	42.870
Average rank of DIEN	50.172	37.007	46.806	43.341

TABLE VI

RANKING OF DIFFERENT MODELS AS THE SIZE OF THE TRAINING DATA INCREASES



(a) Taobao



(b) Retailer-X

Fig. 1. Error analyses on two datasets: the appearance count of an item and the position of the next clicked item in the recommendation list (smaller is better).

value of the position is smaller). This trend is more evident in the Taobao dataset, probably because this dataset contains more highly popular items, so the NN algorithm is able to make better recommendations for these items.

Table VI shows the second result: when we increased the size of the training data, both the DIN model and the DIEN model indeed make better recommendations. Thus, the previous papers reported superior performance for the DIN and DIEN models likely because they applied these models on a large training dataset. As we are unable to obtain the datasets used in these papers, the Taobao dataset on Tianchi is the best we could obtain, but this dataset contains only selected users' logs for nine days. Another aspect that may cause this difference is that the authors who proposed DIN and DIEN applied these models online for A/B testing. Unfortunately, we are unable to access the production system of Taobao, so online A/B testings are impossible.

V. DISCUSSION

The Internet giants have proposed many new recommendation algorithms, and these algorithms mostly rely on deep learning models. However, according to a 2018 report by eMarketer,[5] more than 150 billion dollars is not spent at the top online retailers, so investigating recommendation algorithms that could work on smaller datasets is still valuable and needed. We showed that when the training dataset is small or when the occurrence frequency of an item is low in the training data, these complex deep learning models may perform worse than simple models, such as KNN. Consequently, for medium- and small-sized e-commerce companies with less training instances, it might be more useful to apply simple recommendation models that cost less computational resources and likely perform better on average, or perhaps to use a hybrid strategy.

While we tried to replicate the experiments presented in previous publications, many practical issues limited us from duplicating their settings. These limitations include the availability of the experimental data, the experimental environment (e.g., online A/B testing), hyperparameter selection, and many more. These inevitable differences may cause a gap between the numbers reported by our experiment and those of previous papers. To eliminate the difficulties for future researchers to repeat our experiments, we have made the experimental code available for public use.[6]

For ongoing and future work, we will continue to investigate how medium- and small-sized e-commerce companies may leverage and adjust the research outputs from Internet giants. We are also working on hybrid or ensemble strategies.

REFERENCES

[1] T. Huang, Z. Zhang, and J. Zhang, "FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction," in *Proceedings of the 13th ACM Conference on Recommender Systems*. Association for Computing Machinery, 2019, pp. 169–177.

[2] Y. Feng, F. Lv, W. Shen, M. Wang, F. Sun, Y. Zhu, and K. Yang, "Deep session interest network for click-through rate prediction," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. ijcai.org, 2019, pp. 2301–2307.

[3] B. Liu, R. Tang, Y. Chen, J. Yu, H. Guo, and Y. Zhang, "Feature generation by convolutional neural network for click-through rate prediction," in *The World Wide Web Conference*. Association for Computing Machinery, 2019, pp. 1119–1129.

[5]https://due.com/blog/small-ecommerce-companies-can-thrive/

[6]https://github.com/ncu-dart/Rec-Switch

[4] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33. AAAI Press, 2019, pp. 5941–5948.

[5] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, 2018, pp. 1059–1068.

[6] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. Association for Computing Machinery, 2015, pp. 1809–1818.

[7] H.-H. Chen, "Behavior2vec: Generating distributed representations of users' behaviors on products for recommender systems," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 4, pp. 1–20, 2018.

[8] K. Bi, C. H. Teo, Y. Dattatreya, V. Mohan, and W. B. Croft, "Leverage implicit feedback for context-aware product search," in *Proceedings of the SIGIR 2019 Workshop on eCommerce, co-located with the 42st International ACM SIGIR Conference on Research and Development in Information Retrieval, eCom@SIGIR 2019, Paris, France, July 25, 2019*, ser. CEUR Workshop Proceedings, vol. 2410. CEUR-WS.org, 2019.

[9] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2008, pp. 426–434.

[10] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 995–1000.

[11] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *Proceedings of the 10th ACM Conference on Recommender Systems*. Association for Computing Machinery, 2016, pp. 43–50.

[12] P. Chen and H.-H. Chen, "Accelerating matrix factorization by overparameterization," in *International Conference on Deep Learning Theory and Applications*, 2020.

[13] H.-H. Chen, "Weighted-svd: Matrix factorization with weights on the latent factors," *arXiv preprint arXiv:1710.00482*, 2017.

[14] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. Association for Computing Machinery, 2016, pp. 7–10.

[15] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for CTR prediction," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, C. Sierra, Ed. ijcai.org, 2017, pp. 1725–1731.

[16] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[17] Y. Song, A. M. Elkahky, and X. He, "Multi-rate deep learning for temporal recommendation," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 909–912.

[18] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 729–732.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[20] H.-H. Chen, C.-A. Chung, H.-C. Huang, and W. Tsui, "Common pitfalls in training and evaluating recommender systems," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 37–45, 2017.

[21] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 297–306.

[22] M. F. Dacrema, P. Cremonesi, and D. Jannach, "Are we really making much progress? a worrying analysis of recent neural recommendation approaches," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 101–109.

[23] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 191–198.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[25] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[26] C. Pei, Y. Zhang, Y. Zhang, F. Sun, X. Lin, H. Sun, J. Wu, P. Jiang, J. Ge, W. Ou *et al.*, "Personalized re-ranking for recommendation," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 3–11.

[27] H. Zhu, X. Li, P. Zhang, G. Li, J. He, H. Li, and K. Gai, "Learning tree-based deep model for recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1079–1088.

[28] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li, "Ranking measures and loss functions in learning to rank," in *Advances in Neural Information Processing Systems*, 2009, pp. 315–323.

[29] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.

[30] A. Ng, "Machine learning yearning," *URL: http://www.mlyearning.org/*, 2017.