PIER: Permutation-Level Interest-Based End-to-End Re-ranking Framework in E-commerce

Xiaowen Shi*
Meituan
Beijing, China
shixiaowen03@meituan.com

Fan Yang*
Meituan
Beijing, China
yangfan129@meituan.com

Ze Wang†
Meituan
Beijing, China
wangze18@meituan.com

Xiaoxu Wu
Meituan
Beijing, China
wuxiaoxu04@meituan.com

Muzhi Guan
Meituan
Beijing, China
guanmuzhi@meituan.com

Guogang Liao
Meituan
Beijing, China
liaoguogang@meituan.com

Yongkang Wang
Meituan
Beijing, China
wangyongkang03@meituan.com

Xingxing Wang
Meituan
Beijing, China
wangxingxing04@meituan.com

Dong Wang
Meituan
Beijing, China
wangdong07@meituan.com

ABSTRACT

Re-ranking draws increased attention on both academics and industries, which rearranges the ranking list by modeling the mutual influence among items to better meet users' demands. Many existing re-ranking methods directly take the initial ranking list as input, and generate the optimal permutation through a well-designed context-wise model, which brings the evaluation-before-reranking problem. Meanwhile, evaluating all candidate permutations brings unacceptable computational costs in practice. Thus, to better balance efficiency and effectiveness, online systems usually use a two-stage architecture which uses some heuristic methods such as beam-search to generate a suitable amount of candidate permutations firstly, which are then fed into the evaluation model to get the optimal permutation. However, existing methods in both stages can be improved through the following aspects. As for generation stage, heuristic methods only use point-wise prediction scores and lack an effective judgment. As for evaluation stage, most existing context-wise evaluation models only consider the item context and lack more fine-grained feature context modeling.

This paper presents a novel end-to-end re-ranking framework named *PIER* to tackle the above challenges which still follows the two-stage architecture and contains two mainly modules named FPSM and OCPM. Inspired by long-time user behavior modeling methods, we apply SimHash in FPSM to select top-K candidates from the full permutation based on user's permutation-level interest in an efficient way. Then we design a novel omnidirectional attention mechanism in OCPM to better capture the context information in the permutation. Finally, we jointly train these two modules in an end-to-end way by introducing a comparative learning loss, which use the predict value of OCPM to guide the FPSM to generate better permutations. Offline experiment results demonstrate that PIER outperforms baseline models on both public and industrial datasets, and we have successfully deployed PIER on Meituan food delivery platform.

CCS CONCEPTS

• **Information systems** → **Electronic commerce**.

KEYWORDS

Re-ranking; End-to-End Learning; Recommender Systems

ACM Reference Format:
Xiaowen Shi, Fan Yang, Ze Wang, Xiaoxu Wu, Muzhi Guan, Guogang Liao, Yongkang Wang, Xingxing Wang, and Dong Wang. 2023. PIER: Permutation-Level Interest-Based End-to-End Re-ranking Framework in E-commerce. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3580305.3599886

1 INTRODUCTION

E-commerce applications such as JD.com and Meituan have a large number of items. To improve the user's decision-making efficiency, a slate which contains limited items is usually provided based on the user's interest. As shown in figure 1, a slate named Guess You Like which consists of three items, is displayed to users on Meituan food delivery platform.

Due to the rapid growth of deep learning techniques, many well-designed ranking models have been proposed to improve the recommendation performance, mainly focusing on feature interaction (e.g. Wide&Deep [9], DeepFM [14], xDeepFM [16]), user interest modeling (e.g. DIN [27], DIEN [26], ETA [8]), and so on. However, most existing ranking methods only model the CTR of the current item, but ignore the crucial mutual influence among

*Equal contribution. Listing order is random.
†Corresponding author.

KDD '23, August 6–10, 2023, Long Beach, CA, USA.

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599886

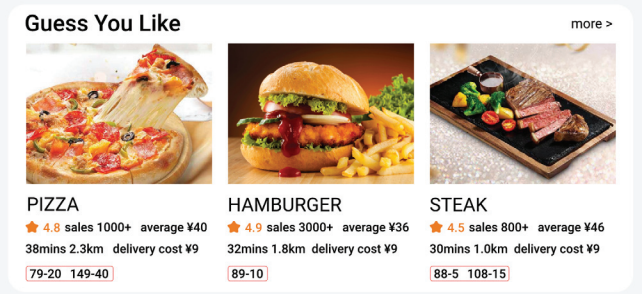

Figure 1: Guess You Like in Meituan.

contextual items. In order to model the influence of the arrangement of displayed items on user behaviors, the re-ranking stage is introduced to rearrange the initial list from the ranking stage.

Existing re-ranking methods can be divided into two categories [12]. The first category is the step-greedy re-ranking strategy [3, 12, 13, 29], which sequentially decides the display results of each position. Such methods only consider the preceding information but ignore the succeeding information, which is insufficient to obtain the optimal result. Different from the greedy strategy, another solution is the context-wise re-ranking strategy [1, 6, 11, 20, 25], which uses a context-wise evaluation model to capture the mutual influence among items and re-predict the CTR of each item. Methods like PRM [20], directly take the initial ranking list as input, and generates the optimal permutation based on the predict value given by context-wise model. These one-stage methods suffer an evaluation-before-reranking problem [25], that is, due to the order obtained after re-ranking is different from the initial order, inputting the re-ranked list will result in different prediction results. In order to resolve evaluation-before-reranking problem, a straightforward solution is to evaluate every possible permutation, which is global-optimal but is too complex to meet the strict inference time constraint in industrial system. Therefore, most existing re-ranking framework uses a two-stage architecture [11, 25] which consists of permutation generation and permutation evaluation. To be specific, they use some heuristic methods such as beam-search [21] to generate the suitable amount of candidate permutations firstly and then feed into the evaluation model to get the optimal permutation.

In order to improve the performance of the re-ranking stage under the two-stage architecture, on the one hand, the generated candidate permutations should contain the optimal permutation as much as possible. On the other hand, the context-wise evaluation model should predict as accurately as possible. However, existing methods in both stages can be improved through the following aspects:

- Generation stage. Some heuristic methods, such as the beam search algorithm [11], merely use point-wise prediction scores (i.e. item CTR) to generate candidate permutations, while ignoring the mutual influence between each item and its contexts in one permutation. In addition, since the generation stage is independent of the evaluation stage, the evaluation results cannot guide the generating process. Therefore, the quality of the generated candidate permutations lacks an effective judgment.
- Evaluation stage. Different from the point-wise item prediction, the evaluation of the permutations needs to use various types of contextual information to fully model the mutual influence among items. In addition to the influence of item context, there is also the fine-grained influence of feature context. These features form a variety of channels, and users may be interested in the features of a certain channel. For example, price-sensitive users will pay more attention to the comparison of price information in the context, which we call it as multi-feature channel competition problem.

To resolve the aforementioned issues, we propose a novel end-to-end re-ranking framework named Permutation-Level Interest-Based End-to-End Re-ranking (PIER). Our framework still follows the two-stage paradigm which contains two mainly modules named Fine-grained Permutation Selection Module (FPSM) and Omnidirectional Context-aware Prediction Module (OCPM). Inspired by long-time user behavior modeling methods [4, 8], we apply SimHash in FPSM to select top-K candidates from the full permutation based on user's permutation-level interest in an efficient way. Then in OCPM, we design a novel omnidirectional attention and context-aware predict mechanism to better capture the context information and predict the list-wise CTR of each item in the permutation. Finally, we integrate these two modules into one framework and training in an end-to-end way. We introduce a comparative learning loss, which use the predict value of OCPM to guide the FPSM to generate better permutations.

The main contributions of our work are summarized as follows:

- We propose a novel re-ranking framework named PIER, which integrates generation module and evaluation module into one model and can be trained in an end-to-end manner.
- We conduct extensive offline experiments on both public dataset and real-world industrial dataset from Meituan. Experimental results demonstrate the effectiveness of PIER. It is notable that PIER has been deployed in Meituan food delivery platform and has achieved significant improvement under various metrics.

2 RELATED WORK

An industrial recommender system typically consists of three stages[1]: matching [28], ranking [15, 24] and re-ranking [20]. Matching stage aims to recall thousands of relevant items from the whole item set. Ranking stage [9, 14, 27] point-wisely predict the click-through rate (or conversion rate, etc.) of recalled items. Re-ranking stage aims to find the best (e.g. maximization of the total clicks) permutation from the initial list given by the ranking model. In this paper, we mainly focus on re-ranking stage.

Typical re-ranking methods can be divided into two categories. The first category is the step-greedy re-ranking strategy [3, 12, 13, 29], which sequentially decides the display results of each position through recurrent neural network or approximation solution. Seq2slate [3] utilizes pointer-network and MIRNN utilizes [29] GRU to determine the item order one-by-one. Similarly, the client-side short video re-ranking framework of Kuaishou [13] combines a point-wise prediction model which takes ordered candidates list as input with beam-search to sequentially generate the final list.

[1]We do not discuss advertising-related stages (e.g. mix-ranking [7, 18], auctions [17]) in this paper.

These methods ignore succeeding information, which is insufficient to obtain the optimal result.

Another category is context-wise re-ranking strategy [1, 6, 11, 20, 25], which uses a context-wise evaluation model to capture the mutual influence among items and re-predict the CTR/CVR of each item. Methods such as PRM [20] and DLCM [1] take the initial ranking list as input, use RNN or self-attention to model the context-wise signal and output the predict value of each item. The optimal permutation is sorted according to the predict value. Such methods bring an evaluation-before-reranking problem [25] and leads to sub-optimum. Similarly, methods such as EXTR [6] estimate pCTR of each candidate item on each candidate position, which are substantially point-wise models and thus limited in extracting exact context. In order to model the exact context of the permutation, a straightforward solution is to evaluate every possible permutation through a well-designed context-wise model. This is a global-optimal method but is too complex to meet the strict inference time constraint in industrial system. In order to reduce the complexity, PRS [11] adopts beam-search to generate few candidate permutations firstly, and score each permutation through a permutation-wise ranking model. Although heuristic methods are effective, they only use point-wise prediction scores and lack an effective judgment, which can be improved further.

In our scenario, we still adopt the two-stage architecture and focus on improving the overall performance by optimizing both two stages under the online time-consuming constraints.

3 PRELIMINARIES

Typically, an industrial recommender system consists of three consecutive stages: matching, ranking and re-ranking [11, 15, 20, 24, 25, 28]. Given a certain user involving his/her input ranking list $C = \{a_i\}_{i=1}^{N_o}$, the final displayed N_d items to user are formulated as the displayed list $\mathcal{P} = \{a_i\}_{i=1}^{N_d}$, where $N_d \leq N_o$.

If there is no re-ranking stages, the top-N_d items are selected as $\mathcal{G}$ from C for display. Therefore, the task of re-ranking stage is to learning a re-ranking strategy $\pi : C \rightarrow \mathcal{P}^*$, which aims to select and rearrange items from C, and subsequently recommends a better displayed list $\mathcal{P}^*$, with the aim of improving indicators such as CTR and GMV.

4 METHODOLOGY

We present the overview structure of PIER (**P**ermutation-level **I**nterest-based **E**nd-to-End **R**e-ranking) in Figure 2. Specifically, we take ranking list $C = \{a_i\}_{i=1}^{N_o}$ and user's permutation-level click behavior sequence $\mathcal{B} = \{b_i\}_{i=1}^{M}$ as the input of PIER. Then, we generate candidate permutations $\mathcal{G} = \{p_i\}_{i=1}^{T}$ on C through full permutation algorithm[2]. Next, we use the Fine-grained Permutation Selection Module (FPSM) to select top-K permutations from large number of candidate permutations. Finally, we use the Omnidirectional Context-aware Prediction Module (OCPM) to calculate scores of each permutation and select the best permutation p^* as the re-ranking list to display.

In FPSM, we propose the time-aware hamming distance calculated based on SimHash [5, 8, 19]. The top-K permutations are selected by sorting the distance between user's permutation-level click behaviors and candidate permutations. In OCPM, we design a novel omnidirectional attention unit to model the context information in each permutation and output the list-wise pCTR of each item in this permutation. The best permutation is selected based on the output score, i.e., the sum of list-wise pCTRs. The relationship between FPSM and OCPM is like the matching and ranking stages in recommender systems. We combine the two into one framework to generate the optimal re-ranking list. Next we will detail FPSM and OCPM separately.

4.1 Fine-grained Permutation Selection Module

For time-consuming considerations, some re-ranking methods utilize heuristic methods such as beam-search [11] to generate candidate permutations and use a well-designed prediction model to select the optimal permutation, but these heuristic methods are not consistent with the modeling target, leading to suboptimal performance. Inspired by long-term user behavior modeling methods such as ETA and SDIM [4], we propose FPSM to select top-K candidates through SimHash. Here, We use the user's historical click behaviors as target and then calculate the distance between it and all candidate permutations. If the distance is closer, we believe that the permutation can better match user's interest thus can bring greater revenue. In this manner, we can not only reduce the time

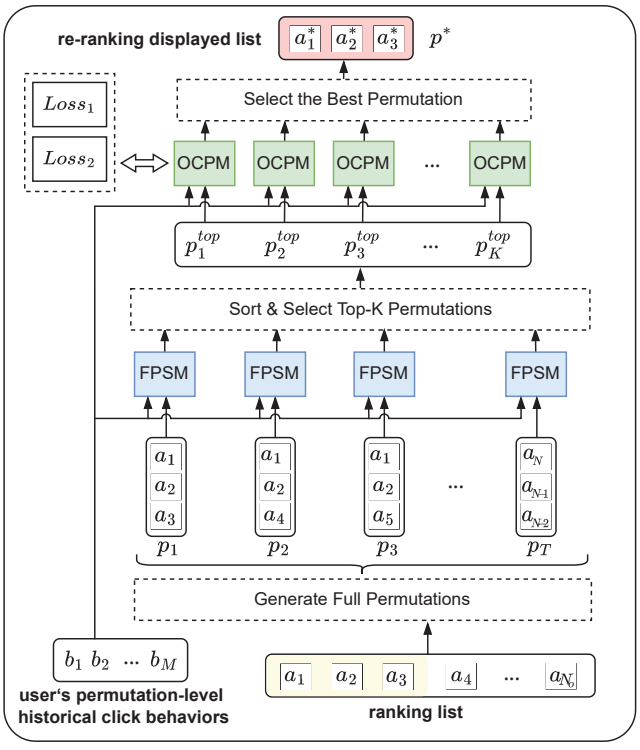

Figure 2: Overview of our framework PIER. PIER takes ranking list and user's permutation-level historical click behaviors as input, and outputs a re-ranking list to display with the help of FPSM and OCPM.

[2]We set the number of items N_d in each permutation to 3 for illustration in figures.

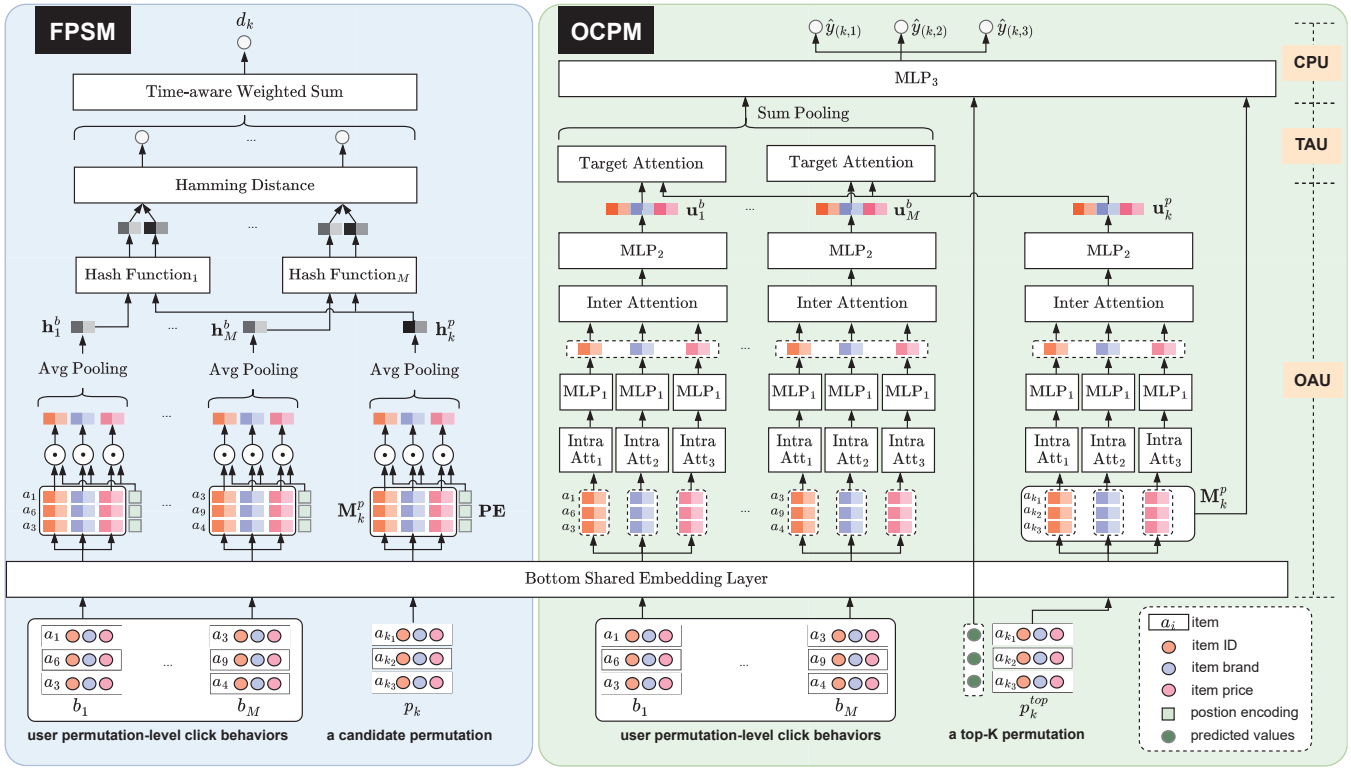

Figure 3: The structures of Fine-grained Permutation Selection Module (FPSM) and Omnidirectional Context-aware Prediction Module (OCPM). FPSM takes user permutation-level click behaviors and a candidate permutation as input and outputs the distance score of this permutation. OCPM takes user permutation-level click behaviors and a top-K permutation selected by FPSM as input and outputs the predicted list-wise pCTR of each item in this permutation. Best view in color.

complexity, but also can make consistent selection by training the FPSM and the prediction model in an end-to-end way. Next, we will introduce how to select the top-K permutations through FPSM.

As shown in the left part of Figure 3, we first use the bottom shared embedding layers to extract the embeddings from raw inputs. For permutation p_k, we denote the embedding matrix $\mathbf{M}_k^p$ as follows:

$$\begin{aligned}\mathbf{M}_k^p &= \begin{bmatrix} \mathbf{E}_{0;}^{p_k} \\ \mathbf{E}_{1;}^{p_k} \\ \cdots \\ \mathbf{E}_{N_d;}^{p_k} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{E}_{;0}^{p_k} & \mathbf{E}_{;1}^{p_k} & \cdots & \mathbf{E}_{;N_f}^{p_k} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{e}_{0;0}^{p_k} & \mathbf{e}_{0;1}^{p_k} & \cdots & \mathbf{e}_{0;N_f}^{p_k} \\ \mathbf{e}_{1;0}^{p_k} & \mathbf{e}_{1;1}^{p_k} & \cdots & \mathbf{e}_{1;N_f}^{p_k} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{e}_{N_d;0}^{p_k} & \mathbf{e}_{N_d;1}^{p_k} & \cdots & \mathbf{e}_{N_d;N_f}^{p_k} \end{bmatrix} \in \mathbb{R}^{N_d \times N_f \times D}, \end{aligned} \tag{1}$$

where N_d is the number of items in each permutation, N_f is the number of feature fields (i.e., ID, category, brand and so on) in each item, D is the dimension of the embedding-transformed feature field, $\mathbf{e}_{i;j}^{p_k} \in \mathbb{R}^D$ is the embedding of the i-th item's j-th feature field in permutation p_k, $\mathbf{E}_{i;}^{p_k} \in \mathbb{R}^{N_f \times D}$ is the embedding matrix of the i-th item in permutation p_k, and $\mathbf{E}_{;j}^{p_k} \in \mathbb{R}^{N_d \times D}$ is the embedding matrix of the j-th feature field in permutation p_k. Analogously, the embedding matrix of the m-th permutations in user's permutation-level history click behaviors is formulated as $\mathbf{M}_m^b \in \mathbb{R}^{N_d \times N_f \times D}$.

Next, we generate the position encoding matrix $\mathbf{PE} \in \mathbb{R}^{N_d \times D}$ for each permutation as follows:

$$\begin{aligned} \mathrm{PE}_{(i,2d)} &= \sin(i/10000^{2d/D}), \\ \mathrm{PE}_{(i,2d+1)} &= \cos(i/10000^{2d/D}), \\ \mathbf{PE} &= \begin{bmatrix} \mathrm{PE}_{(0,0)} & \mathrm{PE}_{(0,1)} & \cdots & \mathrm{PE}_{(0,D)} \\ \mathrm{PE}_{(1,0)} & \mathrm{PE}_{(1,1)} & \cdots & \mathrm{PE}_{(1,D)} \\ \cdots & \cdots & \cdots & \cdots \\ \mathrm{PE}_{(N_d,0)} & \mathrm{PE}_{(N_d,1)} & \cdots & \mathrm{PE}_{(N_d,D)} \end{bmatrix} \in \mathbb{R}^{N_d \times D}. \end{aligned} \tag{2}$$

Then, the embedding matrices of each feature field are multiplied by the position encoding matrix $\mathbf{PE}$ respectively and then merged into corresponding permutation representation $\mathbf{h}_k^p$ by average pooling, as follows:

$$\mathbf{h}_k^p = \frac{1}{N_f} \sum_{i=1}^{N_f} \text{Avg-Pool}\left(\mathbf{E}_{;i}^{p_k} \odot \mathbf{PE}\right), \quad \forall k \in [N_o]. \tag{3}$$

Analogously, the representation of m-th permutation in user's permutation-level history click behaviors is formulated as $\mathbf{h}_m^b$.

In our scenario, users are more likely to click the permutations which are closer to the user's interest. We use user's permutation-level history click behaviors to represent user's interest and calculate the distance between user's interest and each candidate permutation. Specifically, we utilize the random projection schema (SimHash) [5, 8, 19] to calculate the similarity between the representations of user clicked permutations and the representations of candidate permutations. We first generate M different hash functions corresponding to M user's permutation-level behaviors. For each candidate permutation p_k, we hash its representations $\mathbf{h}_k^p$ with M different hash functions and calculate the hamming distance between it and each user's permutation-level behavior, as follows:

$$\mathrm{Sim}(p_k, b_m) = \mathrm{Hash}_m(\mathbf{h}_k^p, \mathbf{h}_m^b), \quad \forall m \in [M], \forall k \in [N_o]. \tag{4}$$

Meanwhile, the more recent behavior, the more it can reflect the user's current interest and will be given a higher weight in similarity calculation. So we weight these distance according to the occurrence time of each behavior to obtain the time-aware hamming distance, as follows:

$$d_k = \sum_{m=1}^{M} w_m \cdot \mathrm{Sim}(p_k, b_m), \quad \forall k \in [N_o]. \tag{5}$$

where w_m is the time-aware weight of m-th behavior.

Finally, we sort candidate permutations based on their distance scores and select top-K permutations $\mathcal{P}^{top-K} = \{p_i^{top}\}_{i=1}^K$ with the smallest distance scores as the output of FPSM.

Since FPSM shares bottom embedding layers with OCPM and fixes random vectors for hash function and position encoding, it dose not have its own independent parameters that need to be trained. In order to ensure the quality of the selected top-K permutations during training, we propose a contrastive loss to improve the performance of FPSM. The detail of contrastive loss is discussed in Section 4.3.2.

4.2 Omnidirectional Context-aware Prediction Module

For each candidate permutation selected by FPSM, we then use OCPM to predict the pCTR of each item through three consecutive unit: Omnidirectional Attention Unit (OAU), Target Attention Unit (TAU), Context-aware Prediction Unit (CPU). The architecture of OCPM is shown in the right part of Figure 3 and we will introduce each unit of OCPM separately.

4.2.1 Omnidirectional Attention Unit (OAU). When showing a permutation to users, on the one hand, they may pay attention to different displayed information such as price, delivery fee, rating, etc. On the other hand, the competitive relationship between the same feature of different items will also affect the user's behavior. For instance, placing expensive items ahead cheap items can stimulate user's desire to click on cheap one. From this point of view, in OCPM, we design an omnidirectional attention unit to effectively modeling the information of each permutation.

Specifically, OCPM first uses the same bottom shared embedding layers as FPSM to extract embeddings from raw inputs. Then we use N_f parameter-independent self-attention layers [22] to calculate the mutual influence of different items in each field separately and output corresponding matrix $\mathbf{H}_j^{p_k}$, as follows:

$$\mathbf{H}_j^{p_k} = \mathrm{soft\,max}(\frac{\mathbf{Q}_j^{p_k}\mathbf{K}_j^{p_k\top}}{\sqrt{D}})\mathbf{V}_j^{p_k}, \quad \forall j \in [N_f], \forall k \in [K], \tag{6}$$

where $\mathbf{Q}_j^{p_k}, \mathbf{K}_j^{p_k}, \mathbf{V}_j^{p_k}$ represent query, key, and value for the j-th field in the k-th target permutation, respectively. D denotes feature dimension of each feature. Here, query, key and value are transformed linearly from $\mathbf{E}_{:,j}^{p_k}$, as follows:

$$\mathbf{Q}_j^{p_k} = \mathbf{E}_{:,j}^{p_k}\mathbf{W}_j^Q, \mathbf{K} = \mathbf{E}_{:,j}^{p_k}\mathbf{W}_j^K, \mathbf{V} = \mathbf{E}_{:,j}^{p_k}\mathbf{W}_j^V, \forall j \in [N_f], \forall k \in [K], \tag{7}$$

where $\mathbf{W}_j^Q, \mathbf{W}_j^K, \mathbf{W}_j^V \in \mathbb{R}^{D\times D}$. Then $\mathbf{H}_j^{p_k}$ are input into a Multi-Layer Perceptrons (MLP) layer to generate representation:

$$\begin{aligned} \mathbf{h}_j^{p_k} &= \mathrm{MLP}_1\left(\mathbf{H}_j^{p_k}\right), \quad \forall j \in [N_f], \forall k \in [K], \\ \mathbf{Z}^{p_k} &= \left[\mathbf{h}_1^{p_k} \quad \mathbf{h}_2^{p_k} \quad \dots \quad \mathbf{h}_{N_f}^{p_k}\right] \in \mathbb{R}^{N_f\times D}, \quad \forall k \in [K], \end{aligned} \tag{8}$$

After that, an inter-field self-attention layer is introduced to calculate the mutual influence between different fields in each permutation and output the final representation of the permutation $\mathbf{u}_k^p$ by an MLP layer, as follows:

$$\mathbf{u}_k^p = \mathrm{MLP}_2\left(\mathbf{H}_j^{p_k}\right) = \mathrm{MLP}_2\left(\mathrm{soft\,max}(\frac{\mathbf{Q}'_{p_k}\mathbf{K}'^{\top}_{p_k}}{\sqrt{D}})\mathbf{V}'_{p_k}\right), \forall k \in [K], \tag{9}$$

where $\mathbf{Q}'_{p_k}, \mathbf{K}'_{p_k}, \mathbf{V}'_{p_k}$ represent query, key, and value for the k-th permutation and are transformed linearly from $\mathbf{Z}^{p_k}$, as follows:

$$\mathbf{Q}'_{p_k} = \mathbf{Z}^{p_k}\mathbf{W}^{Q'}, \mathbf{K}'_{p_k} = \mathbf{Z}^{p_k}\mathbf{W}^{K'}, \mathbf{V}'_{p_k} = \mathbf{Z}^{p_k}\mathbf{W}^{V'}, \forall k \in [K]. \tag{10}$$

Analogously, the final representation of m-th permutation in user's permutation-level history click sequence is formulated as $\mathbf{u}_m^b$. Through these two self-attention layers, we can effectively model the relationship between different items and different feature fields.

4.2.2 Target Attention Unit (TAU). Zhou et al. [27] have proved that historical behaviors, which are more relevant to the target, can provide more information for the model's predict. We use a permutation-level target attention unit to model the interactions between target permutation and each permutation in historical behaviors, as follows:

$$\begin{aligned} \mathbf{w}_{m;k} &= \mathbf{u}_m^b \cdot \mathrm{MLP}_{\mathrm{Att}}\left(\mathbf{u}_k^p || \mathbf{u}_m^b || (\mathbf{u}_k^p \odot \mathbf{u}_m^b) || (\mathbf{u}_k^p - \mathbf{u}_m^b)\right), \\ \mathbf{w}_k &= \text{Sum-Pool}\left([\mathbf{w}_{1;k}, \dots, \mathbf{w}_{M;k}]\right), \forall m \in [M], \forall k \in [K]. \end{aligned} \tag{11}$$

where $||$ means concatenation. The output representation $\mathbf{w}_k$ of TAU is regarded as the representation of user's permutation-level interest on the target permutation and is used as input of next unit.

4.2.3 Context-aware Prediction Unit (CPU). In Context-aware Prediction Unit, we use a parameter-sharing MLP layer to predict the list-wise pCTR of each item in each permutation. Taking the k-th target permutation as an example, the inputs of CPU consist of four parts: representation of the k-th target permutation $\mathbf{u}_k^p$, user's permutation-level interest on the k-th target permutation $\mathbf{w}_k$, the original predicted values (i.e., point-wise pCTRs of each

item) $\{v_i^{p_k}\}_{i=1}^{N_d}$ and embedding matrix of the k-th target permutation $\mathbf{M}_k^p$. Then the list-wise pCTR of the t-th item in the k-th target permutation is predicted as follows:

$$\hat{y}_{(k,t)} = \sigma\Big(\mathrm{MLP}_3(\mathbf{u}_k^p||\mathbf{w}_k||\{v_i^{p_k}\}_{i=1}^{N_d}||\mathbf{M}_k^p)\Big), \tag{12}$$

where σ is Sigmoid Function. In PIER, the score of each permutation is easily obtained by summing the output list-wise pCTR. However, It should be noticed that by using the framework of PIER, the score of each permutation can be conveniently adjusted according to business needs, such as using GMV instead of CTR.

4.3 Model Training

In training stage, We first use the permutation of real exposures to train the OCPM, then jointly train the two modules by adding a contrastive learning loss.

4.3.1 Pre-training of OCPM. In order to accurately evaluate the selected top-K permutations, we first pre-train the OCPM using the samples collected from online log. The inputs are the permutations which are real displayed and the labels are whether the items in each displayed permutation are clicked. Then the loss of the OCPM is calculated as follows:

$$Loss_1 = \sum_{t=1}^{N_d}\Big(-y_{(i,t)}\log(\hat{y}_{(k,t)}) - (1-y_{(i,t)})\log(1-\hat{y}_{(i,t)})\Big), \tag{13}$$

where subscript i is the index of samples and t is the index of displayed items.

4.3.2 Joint training of PIER. In the joint training phase, since we use fix random vector as 'hash function' and 'position encoding', and the embeddings are shared between FPSM and OCPM, FPSM does not need to update gradients. When the embedding is updated, the hash signature is updated correspondingly. So how to improve the quality of the selected top-K permutations during the training process. Here, we propose a contrastive loss to ensure this goal. The main idea is that the difference between the average pCTR of the selected top-K permutations and the average pCTR of any K permutations in unselected permutations is as large as possible. Therefore, the final loss in the joint training phase is:

$$Loss_2 = -\sum_{k=1,k'\notin[K]}^{K}\Big(\frac{1}{N_d}\sum_{t=1}^{N_d}\hat{y}_{(k,t)} - \frac{1}{N_d}\sum_{t=1}^{N_d}\hat{y}_{(k',t)}\Big)^2, \tag{14}$$

where subscript i is the index of samples and t is the index of displayed items.

Finally, we sample a batch of samples $\mathcal{B}$ from the dataset and update PIER using gradient back-propagation w.r.t. the loss:

$$L(B) = \frac{1}{|\mathcal{B}|}\sum_{\mathcal{B}}\Big(Loss_1 + \alpha\cdot Loss_2\Big), \tag{15}$$

where α is the coefficient to balance the two losses.

5 EXPERIMENTS

In this section, we conducted extensive offline experiments and online A/B test to evaluate the effectiveness of our framework[3]. For offline experiments, we will first compare OCPM with existing baselines and analyze the role of different designs in it. Given the same prediction model, we will next compare the performance of our FPSM with other generative method. Finally, we will verify how different hyper-parameter settings (e.g., K, α) affect the performance of our framework. For online experiments, we will compare our framework with the existing strategy deployed on the Meituan platform using an online A/B test.

[3]The code is publicly accessible at https://github.com/Lemonace/PIER_code.

5.1 Experimental Settings

5.1.1 Dataset. In order to verify the effectiveness of our framework, we conduct sufficient experiments on both public dataset and industrial dataset. For public dataset, we choose Avito dataset. For industrial dataset, we use real-world data collected from Meituan food delivery platform. Table 1 gives a brief introduction of the datasets.

• **Avito**. The public Avito dataset contains user search logs and metadata from avito.ru, which contains more then 36M ads, 1.3M users and 53M search requests. The full features include user search information(e.g., userid, searchid, and search date) and ad information(e.g., adid, categoryid, title and so on). Each search id corresponds to a search page with multiple ads. For each user, We rank his search pages in increasing order based on the search date, and use the first T-1 search pages as the behavior pages, and the ads in the T-th search page as the target ads to be predicted. Here we use the data from 20150428 to 20150514 as the training set and the data from 20150515 to 20150520 as the testing set to avoid data leakage.

• **Meituan**. The industrial Meituan dataset is collected on Meituan food delivery platform during April 2022, which contains user information(e.g., userid, gender, age), ad information(e.g., adid, categoryid, brandid and so on). According to the date of data collection, we divide the dataset into training and test sets with the proportion of 8:2.

5.1.2 Evaluation Metrics. For offline experiments, we use different metrics for different modules. Specifically, we use the widely adopted AUC metric to evaluate the effectiveness of the prediction module and use the following metrics to evaluate the whole framework:

- **HR(Hit Ratio)@1** [2]. For each data, HR is 1 only when the top-K permutations selected by generative methods contains the best permutation.
- **Cost**. Cost means the overall time-consuming for different re-ranking frameworks.

For online experiments, we compare our proposed framework with existing method through CTR, GMV and inference time.

5.1.3 Hyperparameters. The hidden layer sizes of MLP_1, MLP_2, MLP_3 are $(128, 64, 32)$, $(60, 32, 20)$, and $(50, 20)$, respectively. the learning rate is 10^{-3}, the optimizer is Adam and the batch size is 1,024. The α is 0.1, the embedding size is 8 and the length of user

Table 1: Statistics of the datasets.

Dataset	#Requests	#Users	#Ads
Avito	53,562,269	1,324,103	23,562,269
Meituan	230,525,531	3,201,922	98,525,531

Table 2: The experimental results about AUC and Logloss on two datasets.

Model	Avito		Meituan	
	AUC	LogLoss	AUC	LogLoss
DNN	0.6876	0.0483	0.6538	0.1917
DCN	0.6896	0.0483	0.6550	0.1916
PRM	0.7131	0.0481	0.6718	0.1899
EXTR	0.7114	0.0481	0.6704	0.1901
Edge-Rerank	0.7163	0.0479	0.6694	0.1909
OCPM	**0.7320**	**0.0471**	**0.6822**	**0.1891**

behavior sequence is 5. For Avito dataset, the length of ranking list and re-ranking list are both 5, thus the length of full permutation is 120 and **K** is set to 10. For Metuan dataset page, we select 3 items from the initial ranking list which contains 10 items, thus the length of full permutation is 720 and **K** is set to 100.

5.2 Offline Experiments For OCPM

5.2.1 Baselines. We compare OCPM with both point-wise and list-wise representative methods as baselines. We select DNN, DCN as point-wise baselines and PRM, EXTR and Edge-Rerank from Kuaishou as list-wise baselines. A brief introduction of these methods are as follows:

- **DNN** [10]. DNN is a basic deep learning method for CTR prediction, which applies MLP for high-order feature interaction.
- **DCN** [23]. DCN explicitly applies feature crossing at each layer, requires no manual feature engineering, and adds negligible extra complexity to the DNN model.
- **PRM** [20]. PRM adjusts an initial list by appling the self-attention mechanism to capture the mutual influence between items.
- **EXTR** [6]. EXternality TRansformer regards target ad with all slots as query and external items as key&value to model externalities in all exposure situations.
- **Edge-Rerank** [13]. Edge-Rerank combines an on-device ranking model and an adaptive beam search method to generate context-aware re-ranking result.

5.2.2 Performance Comparison. Table 2 summarizes the results of offline experiments. All experiments were repeated 5 times and the averaged results are reported. We have the following observations from the experimental results: i) All re-ranking listwise model (e.g. PRM, EXTR) makes great improvements over point-wise model (e.g. DNN, DCN) by modeling the mutual influence among contextual items, which verifies the impact of context on user clicks behavior. ii) Compare with transformer-based model(e.g. PRM, ETXR), Edge-Rerank also improve the CTR prediction because of they use the history sequence and previous item information. iii) Our proposed OCPM brings 0.0189/0.0104 absolute AUC on Avito/Metuan dataset gains over the state-of-the-art independent baseline which is a significant improvement in industrial recommendation system.

5.2.3 Ablation Study. To explore the effectiveness of different modules in OCPM, we conduct ablation studies on Avito and Meituan dataset. All experiments were repeated 5 times and the averaged AUC is reported:

Table 3: Result of ablation experiment on different parts in re-ranking model.

Model	Avito		Meituan	
	AUC	LogLoss	AUC	LogLoss
OCPM	**0.7320**	0.0483	**0.6822**	0.1917
- OAU	0.7198	0.0483	0.6774	0.1916
- TAU	0.7263	0.0481	0.6786	0.1899

- OCPM (-OAU) blocks the omnidirectional attention module. OAU aims to capture item context and feature context information, which could model the competitive relationship between the same attribute of different item. As shown in Table 3, AUC decreases by 0.0112/0.0048, suggesting that extracting context information is crucial and the proposed OAU meets this requirement.
- OCPM (-TAU) does not use the target attention unit. TAU aims to compute the interactions of target permutation and each permutation in historical behavior. As shown in Table 3, AUC decreases by 0.0057/0.0036 on Avito/Metuan dataset, suggesting that our TAU are capable to capture user history page-level interest.

5.3 Offline Experiments For PIER

5.3.1 Baselines. We compare our whole framework which combines FPSM and OCPM with some heuristic generative methods as baselines. All these methods use a fixed OCPM but the candidate permutations are generated in different ways. A brief introduction of these methods is as follows:

- **Random & OCPM**. We randomly select K permutations as candidates.
- **PRS(Beam Search & OCPM)** [11]. We use the beam search method to generate K candidate permutations based on the cumulative CTR.
- **Full Permutation & OCPM**. We directly feed all candidate permutations into the prediction model and select K permutations based on average CTR. This can be seen as the upper bound of our framework.

5.3.2 Performance Comparison. Table 4 summarizes the results of offline experiments and we have the following observations: i) Intuitively, PIER achieves great improvements over random and beam-search methods on finding the best permutation on both public dataset and industrial dataset with a small increase in time complexity. One reasonable explanation is that FPSM can select better permutations through the guidance of contrastive loss. ii) We still have much room for improvement compared with the full permutation method on finding the best permutation, but we greatly reduce the time cost of the whole framework.

5.3.3 Ablation Study. To verify the impact of different units (i.e., time-aware weighting, contrastive loss), we study two ablated variants of PIER framework:

- PIER (-time-aware weighting) does not use the time-aware weight on each behavior and treat them as equally important.
- PIER (-contrastive Loss) removes the contrastive loss for improving the quality of the selected top-K permutations, i.e., $\alpha = 0$.

Table 4: HR and Cost of different methods.

Model	Avito		Meituan	
	HR	Cost (ms)	HR	Cost (ms)
Random & OCPM	0.09	**10.8**	0.13	**70.5**
PRS (Beam-Search & OCPM)	0.62	16.8	0.72	78.7
Full Permutation & OCPM	1.00	100.5	1.00	430.5
PIER (FPSM & OCPM)	**0.78**	17.5	**0.84**	85.3
- time-aware weighting	0.74	17.3	0.70	84.8
- contrastive loss	0.52	17.5	0.61	85.1

Since removing a certain unit requires re-training the entire framework, we try to ensure the AUC of OCPM are as close as possible to guarantee the comparability of the ablation experiment. Judging from the experimental results, we have the following findings: i) The performance gap between w/ and w/o time-aware weighting on distance calculation has little impact on public dataset but the performance on industrial dataset is affected to some extent. This indicates that in the Meituan scenario, users' interests remain consistent in a short period of time. ii) The decline in the performance without contrastive loss is obvious on both dataset. This indicates that the auxiliary loss enables FPSM to select better permutations.

Table 5: Result of ablation experiment on different parts in PIER

Settings	Avito			Meituan		
	AUC	HR	Cost (ms)	AUC	HR	Cost (ms)
$\alpha = 0$	0.7337	0.52	17.5	0.6833	0.57	85.1
$\alpha = 0.01$	0.7336	0.63	17.4	0.6831	0.62	85.3
$\alpha = 0.05$	0.7329	0.71	17.6	0.6828	0.71	85.3
$\alpha = 0.1$	0.7320	0.78	17.5	0.6822	0.84	85.2
$\alpha = 0.3$	0.7108	0.81	17.4	0.6674	0.88	85.4
$\alpha = 0.5$	0.6927	0.88	17.6	0.6425	0.91	85.3
$K = 5$	0.7325	0.63	13.8	-	-	-
$K = 10$	0.7320	0.71	17.5	-	-	-
$K = 20$	0.7318	0.82	25.2	-	-	-
$K = 50$	0.7299	0.93	46.9	0.6827	0.78	49.2
$K = 100$	-	-	-	0.6822	0.84	85.3
$K = 200$	-	-	-	0.6813	0.92	157.3
$K = 300$	-	-	-	0.6799	0.95	223.3

5.4 Hyperparameter Analysis

We analyze the sensitivity of two hyperparameters: α, K. Specifically, α is the weight of the contrastive loss and K is number of permutations selected by FPSM. The result is shown in table 5, showing the same trend on public dataset and industrial dataset and we have the following findings :

i) As α increases within a certain range, the AUC of OCPM maintains a relatively good level, while HR is improved. When α exceeds a certain level, the influence of the contrastive loss on OCPM increases, and the AUC begins to decline, resulting in a decrease in the confidence of the HR indicator.

ii) Changing K mainly affects HR and average cost of the framework. Increasing K within a certain range can quickly improve the HR performance. When it exceeds a certain range, HR increases slowly. Meanwhile, the average cost increases with the increase of K. Therefore, the value of K should be set reasonably to balance the effect and efficiency in practice.

Table 6: The experimental results from Online A/B testing.

Model	CTR	GMV	Cost(ms)	Time-out
Base Ranking Model	0%	0%	**0**	0.0%
PRM	1.21%	0.92%	6.1	0.0%
Beam-Search & Evaluator	3.17%	2.54%	10.5	0.052%
Full Permutation & Evaluator	-	-	52.4	64.39%
PIER	**5.46%**	**5.83%**	11.1	**0.061%**

5.5 Online Results

We compare PIER with other models (e.g. base ranking model, PRM and so on) and all deployed on Meituan food delivery platform through online A/B test. Specifically, we conduct online A/B test with 1% of whole production traffic from April 09, 2022 to April 25, 2022 (one week). As a result, we find that PIER gets CTR and GMV increase by 5.46% and 5.83% respectively. Besides, we focus on time costs, which is an important indicator, determine whether it can be applied to a large scale of industrial scenarios. As show in Tabel 6, Full-Permutation could not deploy because of time-out ratio increase by 64.39%. Compared with beam-search, PIER has improved CTR by 2.29% and GMV by 2.31% while the time-out ratio effect increases little, which is acceptable to the system. Now, PIER has been deployed online and serves the main traffic, and contributes to significant business growth.

6 CONCLUSIONS

This paper presents a novel end-to-end re-ranking framework named PIER which contains two mainly modules named FPSM and OCPM. Inspired by long-time user behavior modeling methods, we apply SimHash in FPSM to select top-K candidates from the full permutation. For better capturing the context information in the permutation, we design a novel omnidirectional attention mechanism in OCPM. Finally, we jointly train these two modules in an end-to-end way by introducing a comparative learning loss to guide the FPSM to generate better permutations. Both offline experiment and online A/B test show that PIER significantly outperformed other existing re-ranking baselines, and we have deployed PIER on Meituan food delivery platform.

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 135–144.

[2] Areej Alsini, Du Q Huynh, and Amitava Datta. 2020. Hit ratio: An evaluation metric for hashtag recommendation. *arXiv preprint arXiv:2010.01258* (2020).

[3] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2018. Seq2slate: Re-ranking and slate optimization with rnns. *arXiv preprint arXiv:1810.02019* (2018).

[4] Yue Cao, Xiaojiang Zhou, Jiaqi Feng, Peihao Huang, Yao Xiao, Dayao Chen, and Sheng Chen. 2022. Sampling Is All You Need on Modeling Long-Term User Behaviors for CTR Prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2974–2983.

[5] Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th annual ACM symposium on Theory of computing*. 380–388.

[6] Chi Chen, Hui Chen, Kangzhi Zhao, Junsheng Zhou, Li He, Hongbo Deng, Jian Xu, Bo Zheng, Yong Zhang, and Chunxiao Xing. 2022. EXTR: Click-Through Rate Prediction with Externalities in E-Commerce Sponsored Search. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2732–2740.

[7] Dagui Chen, Qi Yan, Chunjie Chen, Zhenzhe Zheng, Yangsu Liu, Zhenjia Ma, Chuan Yu, Jian Xu, and Bo Zheng. 2022. Hierarchically Constrained Adaptive Ad Exposure in Feeds. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3003–3012.

[8] Qiwei Chen, Changhua Pei, Shanshan Lv, Chao Li, Junfeng Ge, and Wenwu Ou. 2021. End-to-End User Behavior Retrieval in Click-Through Rate Prediction Model. *arXiv preprint arXiv:2108.04468* (2021).

[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[10] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[11] Yufei Feng, Yu Gong, Fei Sun, Junfeng Ge, and Wenwu Ou. 2021. Revisit recommender system in the permutation prospective. *arXiv preprint arXiv:2102.12057* (2021).

[12] Yufei Feng, Binbin Hu, Yu Gong, Fei Sun, Qingwen Liu, and Wenwu Ou. 2021. GRN: Generative Rerank Network for Context-wise Recommendation. *arXiv preprint arXiv:2104.00860* (2021).

[13] Xudong Gong, Qinlin Feng, Yuan Zhang, Jiangling Qin, Weijie Ding, Biao Li, Peng Jiang, and Kun Gai. 2022. Real-time Short Video Recommendation on Mobile Devices. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3103–3112.

[14] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 1725–1731.

[15] Xiang Li, Shuwei Chen, Jian Dong, Jin Zhang, Yongkang Wang, Xingxing Wang, and Dong Wang. 2023. Decision-Making Context Interaction Network for Click-Through Rate Prediction. *arXiv preprint arXiv:2301.12402* (2023).

[16] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 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*. 1754–1763.

[17] Guogang Liao, Xuejian Li, Ze Wang, Fan Yang, Muzhi Guan, Bingqi Zhu, Yongkang Wang, Xingxing Wang, and Dong Wang. 2022. NMA: Neural Multi-slot Auctions with Externalities for Online Advertising. *arXiv preprint arXiv:2205.10018* (2022).

[18] Guogang Liao, Ze Wang, Xiaoxu Wu, Xiaowen Shi, Chuheng Zhang, Yongkang Wang, Xingxing Wang, and Dong Wang. 2022. Cross dqn: Cross deep q network for ads allocation in feed. In *Proceedings of the ACM Web Conference 2022*. 401–409.

[19] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting Near-Duplicates for Web Crawling. In *Proceedings of the 16th International Conference on World Wide Web*. https://doi.org/10.1145/1242572.1242592

[20] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 3–11.

[21] D Raj Reddy et al. 1977. Speech understanding systems: A summary of results of the five-year research effort. *Department of Computer Science. Camegie-Mell University, Pittsburgh, PA* 17 (1977), 138.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[23] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[24] Zhe Wang, Liqin Zhao, Biye Jiang, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2020. Cold: Towards the next generation of pre-ranking system. *arXiv preprint arXiv:2007.16122* (2020).

[25] Yunjia Xi, Weiwen Liu, Xinyi Dai, Ruiming Tang, Weinan Zhang, Qing Liu, Xiuqiang He, and Yong Yu. 2021. Context-aware reranking with utility maximization for recommendation. *arXiv preprint arXiv:2110.09059* (2021).

[26] 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 *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5941–5948.

[27] Guorui Zhou, Xiaoqiang Zhu, Chenru 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 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[28] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1079–1088.

[29] Tao Zhuang, Wenwu Ou, and Zhirong Wang. 2018. Globally optimized mutual influence aware ranking in e-commerce search. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3725–3731.