Revisit Recommender System in the Permutation Prospective

Yufei Feng, Yu Gong, Fei Sun, Qingwen Liu, Wenwu Ou
Alibaba Group, Hangzhou, China
fyf649435349@gmail.com,{gongyu.gy,ofey.sf,xiangsheng.lqw,santong.oww}@alibaba-inc.com

ABSTRACT

Recommender systems (RS) work effective at alleviating information overload and matching user interests in various web-scale applications. Most RS retrieve the user's favorite candidates and then rank them by the rating scores in the greedy manner. In the permutation prospective, however, current RS come to reveal the following two limitations: 1) They neglect addressing the permutation-variant influence within the recommended results; 2) Permutation consideration extends the latent solution space exponentially, and current RS lack the ability to evaluate the permutations. Both drive RS away from the permutation-optimal recommended results and better user experience.

To approximate the permutation-optimal recommended results effectively and efficiently, we propose a novel permutation-wise framework PRS in the re-ranking stage of RS, which consists of Permutation-Matching (PMatch) and Permutation-Ranking (PRank) stages successively. Specifically, the PMatch stage is designed to obtain the candidate list set, where we propose the FPSA algorithm to generate multiple candidate lists via the permutation-wise and goal-oriented beam search algorithm. Afterwards, for the candidate list set, the PRank stage provides a unified permutation-wise ranking criterion named LR metric, which is calculated by the rating scores of elaborately designed permutation-wise model DPWN. Finally, the list with the highest LR score is recommended to the user. Empirical results show that PRS consistently and significantly outperforms state-of-the-art methods. Moreover, PRS has achieved a performance improvement of 11.0% on PV metric and 8.7% on IPV metric after the successful deployment in one popular recommendation scenario of Taobao application.

KEYWORDS

Recommender System; Re-ranking; Permutation-wise

ACM Reference Format:
Yufei Feng, Yu Gong, Fei Sun, Qingwen Liu, Wenwu Ou. 2021. Revisit Recommender System in the Permutation Prospective. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recommender systems (RS) have been widely deployed in various web-scale applications, including e-commerce [10, 14, 28, 34], social media [10, 17] and news [11, 23, 24]. Typically, web-scale RS usually consist of three stages successively, i.e., matching, ranking, and re-ranking. With the continuous advances in the recommendation technologies, various methods have been proposed in each stage to improve user experience and recommendation performance. Overall, most methods in RS contribute to retrieve the user's favorite items from tremendous candidates, followed by the greedy strategy based on the rating scores of given user-item pairs.

Different from point-wise methods in the matching and ranking stages, various list-wise methods [1, 25, 26, 35] have been proposed and deployed in the re-ranking stage. The widely adopted pipeline of existing list-wise methods usually contain three key components: 1) *Ranking*: the initial list is generated according to scores of the basic ranking model; 2) *Refining*: the list-wise feature distribution of the initial list is extracted by a well-designed module (*e.g.,* LSTM [1] and self-attention [25, 26]) to refine the rating scores; 3) *Re-ranking*: candidate items are re-ranked by the refined list-wise rating scores in the greedy manner. Overall, existing list-wise methods achieve improvements in recommendations mainly by modeling list-wise feature distribution to focus on refining the item's rating scores.

In the permutation prospective, however, the popular re-ranking pipeline cannot assist current RS to reach permutation-optimal due to neglecting permutation-variant influence within the recommended results. As shown in Fig. 1 (1), here are two recommended results collected from the real-world dataset. Surprisingly, even if they are composed of the *same* items and exhibited to the *same* user, the user responses to the permutation 2 rather than 1. One possible reason is that placing the more expensive item *B* ahead can stimulate the user's desire to buy the cheaper item *A*. We refer such influence on the user's making decisions caused by the

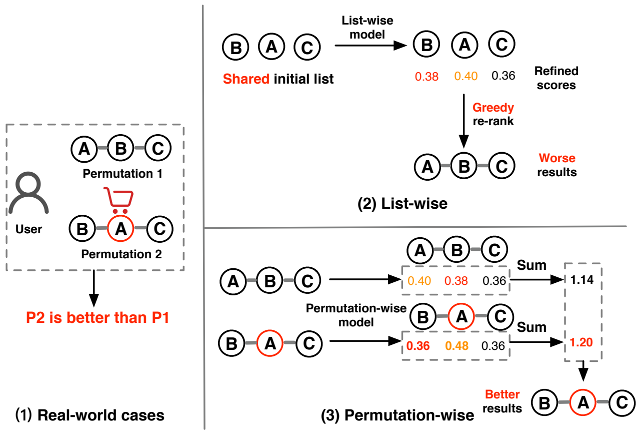

Figure 1: (1) shows two real-world cases. (2) and (3) present the comparison of list-wise and permutation-wise methods when meeting the two cases.

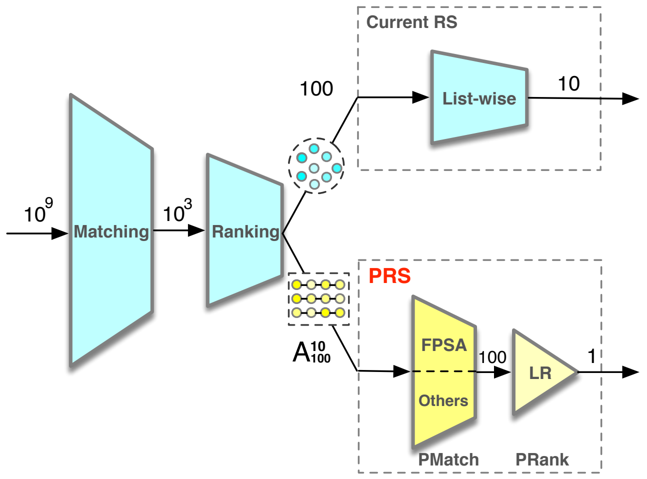

Figure 2: Difference of architecture between current RS and PRS in the re-ranking stage.

change of permutations as *permutation-variant influence* within the recommended results. By addressing such permutation-variant influence, the ideal method should recommend permutation 2 to the user, which is the *permutation-optimal* result in the permutation prospective. While with the supervised training of item A in the permutation 2, the list-wise models in Fig. 1 (2) can only acquire the priority of item A when meeting the shared initial list (*e.g.,* item B, A, C). Afterwards, the list-wise models will re-rank items in the greedy manner by the refined rating scores (*i.e.,* permutation 1) and miss the better permutation 2. Ideally, with the permutation-wise methods in Fig. 1 (3), the predicted interaction probability of item A has been greatly improved by placing item B ahead of A from permutation 1 to 2, which mainly contributes to the correct judgment of permutation 2 better than 1. Accordingly, fully considering and leveraging such permutation-variant influence contained in lists serve as the key to approximate the permutation-optimal recommended results and better user experience.

Practically, such permutation prospective brings new challenges to current RS, which can mainly be summarized as the following two aspects: 1) **Exponential solutions.** Suppose n (*e.g.,* 10) items need to be recommended from the initial list of size m (*e.g.,* 100). Current RS regard it as a retrieval task, that is, deploying list-wise models in the re-ranking stage to search in the O(m) solution space. In the permutation prospective, however, the solution space swells from O(m) to $O(A^n_m)$ (about $O(100^{10})$). Considering the permutation-variant influence within the final item lists, the user will response differently to each list. In fact, only one list will be finally recommended to the user, thus how to reach the permutation-optimal list effectively and efficiently brings new challenges to current RS; 2) **Permutation-wise evaluation.** Most methods applied in current RS attempt to point-wisely predict the user-item interaction probability (*e.g.,* click-through rate and conversion rate). As mentioned above, it is necessary to provide a unified permutation-wise ranking criterion for selecting the permutation-optimal one from massive qualified lists, which has not been involved in the existing efforts.

In this work, as shown in Fig. 2, we promote the list-wise methods to a novel permutation-wise framework named PRS (**P**ermutation **R**etrieve **S**ystem) in the re-ranking stage of RS. Specifically, PRS consists of PMatch (**P**ermutation-**M**atching) and PRank (**P**ermutation-**R**anking) stages successively. The PMatch stage focuses on generating multiple candidate lists in an efficient and parallel way, with the aim of considering permutation-variant influence and alleviating the problem of exponential solutions. Here we propose FPSA (**F**ast **P**ermutation **S**earching **A**lgorithm), a permutation-wise and goal-oriented beam search algorithm, to generate candidate lists in an efficient way. Afterwards, the PRank stage provides a unified ranking criterion for the challenge of permutation-wise evaluation. We equip the proposed model DPWN (**D**eep **P**ermutation-**W**ise **N**etwork) with Bi-LSTM, with which permutation-variant influence can be fully addressed. Moreover, we propose the LR (**L**ist **R**eward) metric to rank the candidate lists, which is calculated by adding the DPWN's rating scores of each item within the candidate list. Finally, the list with the highest LR score will be recommended to the user.

To demonstrate the effectiveness of PRS, we perform a series of experiments on a public dataset from Alimama and a proprietary dataset from Taobao application. Experimental results demonstrate that PRS consistently and significantly outperforms various state-of-the-art methods. Moreover, PRS has been successfully deployed in one popular recommendation scenario of Taobao application and gained a performance improvement of 11.0% on the PV (Page View) metric and 8.7% on the IPV (Item Page View) metric.

2 RELATED WORK

In this section, we review the most related studies in the re-ranking stage in both academic and industrial recommender systems. Typically, the re-ranking stage serves as the last stage after the matching and ranking stages, where an initial ranking list obtained from previous stages is re-ranked into the final item list (*i.e.,* recommended results) to better satisfy user demands. Roughly speaking, existing re-ranking methods mainly fall into three categories: point-wise, pair-wise and list-wise methods. Point-wise methods [9, 10, 19] regard the recommendation task as a binary classification problem and globally learn a scoring function for a given user-item pair with manually feature engineering. Though with continuous achievements, these methods neglect considering the comparison and mutual influence between items in the input ranking list. To solve this problem, pair-wise and list-wise models are gradually paid more and more attention in current RS. Pair-wise models work by considering the semantic distance of an item pair a time. RankSVM [20], GBRank [32] and RankNet [5] apply SVM, GBT and DNN with techinically designed pair-wise loss functions to compare any item pair in the input ranking list, respectively. However, pair-wise methods neglect the local information in the list and increase the model complexity. List-wise models are proposed to capture the interior correlations among items in the list in different ways. LambdaMart [6] is a well-known tree-based method with the list-wise loss function. MIDNN [35] works by handmade global list-wise features, while it requires much domain knowledge and decreases its generalization performance. DLCM [1], PRM [26], and SetRank [25] apply GRU, self-attention, and induced self-attention to encode the list-wise information of the input ranking list for better prediction, respectively. Though effective, this type of list-wise models does not escape the paradigm of greedy ranking based on rating scores, where the permutation-variant influence within recommended results are not fully addressed.

There are also some other works [7, 15, 16, 29] focusing on making the trade-off between relevance and diversity in the re-ranking stage. Another emerging direction of re-ranking methods is group-wise methods [2], where the relevance score of a document is determined jointly by multiple documents in the list. Though effective, group-wise methods may not be appropriate for industrial recommender systems due to its high computation complexity (at least $O(N^2)$).

Note that our proposed PRS can be easily integrated with existing works, deploying them in parallel and merging them into the candidate list set in the PMatch stage.

3 PRELIMINARY

In general, a web-scale recommender system (*e.g.,* e-commerce and news) is composed of three stages chronologically: matching, ranking and re-ranking. In this paper, we focus on the final re-ranking stage, whose input is the ranking list produced by the previous two stage (*i.e.,* matching and ranking). The task of the re-ranking is to elaborately select candidates from the input ranking list and rearrange them into the final item list, followed by the exhibition for users. Mathematically, with user set $\mathcal{U}$ and item set $\mathcal{I}$, we denote labeled list interaction records as $\mathcal{R} = \{(u, \mathcal{C}, \mathcal{V}, \mathcal{Y}^{\text{CTR}}, \mathcal{Y}^{\text{NEXT}} | u \in \mathcal{U}, \mathcal{V} \subset \mathcal{C} \subset \mathcal{I})\}$. Here, $\mathcal{C}$ and $\mathcal{V}$ represent the recorded input ranking list with m items for re-ranking stage and the recorded final item list with n items exhibited to user u, respectively. Intuitively, $n \leq m$. $y_t^{\text{CTR}} \in \mathcal{Y}^{\text{CTR}}$ is the implicit feedback of user u *w.r.t.* t-th item $v_t \in \mathcal{V}$, where $y_t^{\text{CTR}} = 1$ when interaction (*e.g.,* click) is observed, and $y_t^{\text{CTR}} = 0$ otherwise. Similarly, $y_t^{\text{NEXT}} = 1$ indicates that the user continue browsing the followings after this item, and $y_t^{\text{NEXT}} = 0$ otherwise. In the real-world industrial recommender systems, each user u is associated with a user profile x_u consisting of sparse features x_s^u (*e.g.,* user id and gender) and dense features x_d^u (*e.g.,* age), while each item i is also associated with a item profile x_i consisting of sparse features x_i^s (*e.g.,* item id and brand) and dense features x_i^d (*e.g.,* price).

Given the above definitions, we now formulate the re-ranking task to be addressed in this paper:

Definition 1. ***Task Description***. *Typically, the purpose of industrial RS is to make users take more browsing (PV, page view) and interactions (IPV, item page view), and the same for the re-ranking task. Given a certain user* u*, involving his/her input ranking list* $\mathcal{C}$*, the task is to learn a re-ranking strategy* $\pi : \mathcal{C} \xrightarrow{\pi} \mathcal{P}$*, which aims to select and rearrange items from* $\mathcal{C}$*, and subsequently recommend a final item list* $\mathcal{P}$.

4 THE PROPOSED FRAMEWORK

In this section, we introduce our proposed permutation-wise framework PRS, which aims to leverage permutation-variant influence effectively and efficiently for better item rearrangements in the re-ranking stage of RS. To address the exponential solutions and permutation-wise evaluation challenges, we transform the re-ranking task into two successive stages: Permutation-Matching (PMatch) and Permutation-Ranking (PRank). In the PMatch stage, multiple efficient methods can be deployed in parallel to consider permutation-variant influence and search for effective candidate item lists from exponential solutions. Successively, the PRank stage proposes the LR metric as a unified permutation-wise ranking criterion for the candidate item list set, with which the list of the highest LR score will be finally recommended. The key notations we will use throughout the article are summarized in Tab. 1.

Table 1: Key notations.

Notations	Description
$\mathcal{U}, \mathcal{I}$	the set of users and items, respectively
$\mathcal{C}, \mathcal{V}$	the recorded input ranking list and labeled final item list, respectively
$\mathcal{Y}^{\text{CTR}}, \mathcal{Y}^{\text{NEXT}}$	the implicit feedback whether the user clicks or continues browsing of the final item list, respectively
m, n	the number of items in the recorded input and final ranking list, respectively
$\pi, \mathcal{P}$	the learned re-ranking strategy and the generated final ranking list, respectively
$x_s^u(x_s^i), x_d^u(x_s^i)$	the sparse and dense feature set for users (items), respectively
α, β	Fusion coefficient float of r^{PV} and r^{IPV} in FPSA, respectively
r^{PV}, r^{IPV}, r^{sum}	the estimated PV, IPV and mixed reward of the list, respectively
$\mathcal{S}$	the candidate list set generated by the PMatch stage

First of all, we begin with the representations of users and items, which are basic inputs of our proposed framework. Following previous works [10, 34], we parameterize the available profiles into vector representations for users and items. Given a user u, associated with sparse features x_u^s and dense features x_u^d, we embed each sparse feature value into d-dimensional space. Subsequently, each user u can be represented as $\mathbf{x}_u \in \mathbb{R}^{|x_u^s| \times d + |x_u^d|}$, where $|x_u^s|$ and $|x_u^d|$ denote the size of sparse and dense feature space of user u, respectively. Similarly, we represent each item i as $\mathbf{x}_i \in \mathbb{R}^{|x_i^s| \times d + |x_i^d|}$. Naturally, we represent the recorded input ranking list $\mathcal{C}$ as $\mathcal{C} = [\mathbf{x}_c^1, \ldots, \mathbf{x}_c^m]$ and the labeled final item list $\mathcal{V}$ as $\mathcal{V} = [\mathbf{x}_v^1, \ldots, \mathbf{x}_v^n]$, where m and n is the number of items in the input ranking list and final item list, respectively.

In the following sections, we shall zoom into the PMatch and PMatch stages of proposed permutation-wise framework PRS. For each stage, we split the introductions into two parts: one is offline training (preparation) and the other is online serving (inference). The major difference of the two parts is the $\mathcal{V}, \mathcal{Y}^{\text{CTR}}, \mathcal{Y}^{\text{NEXT}}$ in $\mathcal{R}$ is given in the offline training, while not in the online serving.

4.1 Permutation-Matching (PMatch) stage

As motivated, the Permutation-Matching (PMatch) stage is proposed to obtain multiple effective candidate lists in a more efficient

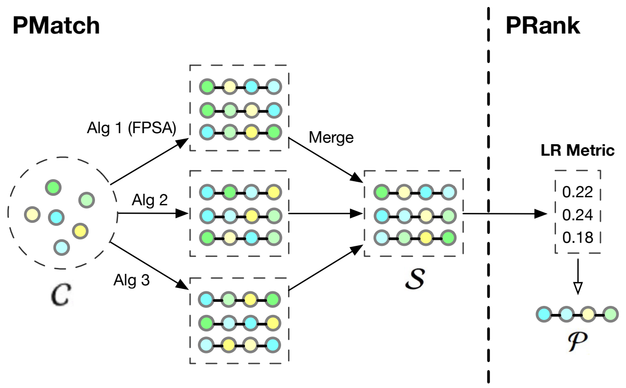

Figure 3: Overall architecture of the PRS framework.

way. As shown in the left part of Fig. 3, various algorithms [1, 25, 26, 35] can be deployed in parallel to generate item lists, followed by the list merge operation into the candidate list set. Though effective, current efforts neglect the permutation-variant influence within the final item list. To leverage the permutation-variant influence in a efficient way, we propose a permutation-wise and goal-oriented beam search algorithm named FPSA (**F**ast **P**ermutation **S**earching **A**lgorithm).

4.1.1 Offline training. To make users take more browsing and interactions, besides the normally applied CTR [14, 34] score, we first design the NEXT score, which predicts the probability whether the user will continue browsing after the item. Items with higher NEXT scores can increase the user's desire for continuous browsing, which improves the probability of subsequent items being browsed and clicked.

Specifically, with the labeled interaction records $\mathcal{R}$, we elaborate on two point-wise models: the CTR prediction model $M^{\text{CTR}}(v|u;\mathbf{\Theta}^{\text{CTR}})$ and the NEXT prediction model $M^{\text{NEXT}}(v|u;\mathbf{\Theta}^{\text{NEXT}})$, which can be calculated as follows:

$$\begin{aligned}\hat{y}_t^{\text{CTR}} &= M^{\text{CTR}}(v|u;\mathbf{\Theta}^{\text{CTR}}) \\ &= \sigma\big(f(f(f(\mathbf{x}_v \oplus \mathbf{x}_u)))\big) \\ \hat{y}_t^{\text{NEXT}} &= M^{\text{NEXT}}(v|u;\mathbf{\Theta}^{\text{NEXT}}) \\ &= \sigma\big(f(f(f(\mathbf{x}_v \oplus \mathbf{x}_u)))\big)\end{aligned} \tag{1}$$

where $f(\mathbf{x}) = ReLU(\mathbf{W}\mathbf{x}+\mathbf{b})$ [1], $\sigma(\cdot)$ is the logistic function. Clearly, the two models can be optimized via binary cross-entropy loss function, which is defined as follows:

$$\begin{aligned}\mathcal{L}^{\text{CTR}} &= -\frac{1}{N}\sum_{(u,\mathcal{V})\in\mathcal{R}}\sum_{x_v^t\in\mathcal{V}}\Big(y_t^{\text{CTR}}\log\hat{y}_t^{\text{CTR}} \\ &\quad + (1-y_t^{\text{CTR}})\log(1-\hat{y}_t^{\text{CTR}})\Big) \\ \mathcal{L}^{\text{NEXT}} &= -\frac{1}{N}\sum_{(u,\mathcal{V})\in\mathcal{R}}\sum_{x_v^t\in\mathcal{V}}\Big(y_t^{\text{NEXT}}\log\hat{y}_t^{\text{NEXT}} \\ &\quad + (1-y_t^{\text{NEXT}})\log(1-\hat{y}_t^{\text{NEXT}})\Big)\end{aligned} \tag{2}$$

We train $\mathbf{\Theta}^{\text{CTR}}$ and $\mathbf{\Theta}^{\text{NEXT}}$ till converged according to Eq.2.

[1]Note that the parameters among different $f(\cdot)$s are not shared.

4.1.2 Online serving. In this work, we regard the re-ranking task as selecting items from the input ranking list sequentially till the pre-defined length. Beam search [21, 30] is a common technology to generate multiple effective lists by exploring and expanding the most promising candidates in a limited set. In the FPSA algorithm, we implement the beam search algorithm in a goal-oriented way, that is, we select the lists with the higher estimated reward at each step.

We clearly present and illustrate the proposed FPSA algorithm in Fig. 4 and Alg. 1. First, we attach two predicted scores to each item c_i in the input ranking list C by the converged CTR prediction model $M^{\text{CTR}}(v|u;\mathbf{\Theta}^{\text{CTR}})$ and the NEXT prediction model $M^{\text{NEXT}}(v|u;\mathbf{\Theta}^{\text{NEXT}})$: the click-through probability $P_{c_i}^{\text{CTR}}$ and the continue-browsing probability $P_{c_i}^{\text{NEXT}}$. Afterwards, at each step we exhaustively expand the remaining candidates for each list in the set $\mathcal{S}$ and reserve the top k of candidate lists according to their calculated estimated reward (lines 1-17). In line 18-28, At each step when iterating through each item in the list, we calculate the transitive expose probability p^{Expose} multiplied by the NEXT scores of previous items. Influenced by p^{Expose}, we calculate PV reward r^{PV} and IPV reward r^{IPV}, and denote the sum of them as the reward r^{sum} at the end of iteration. Here r^{PV}, r^{IPV} and r^{sum} represents the estimated PV, IPV and mixed reward of the list, respectively.

At the end of the algorithm, we obtain the candidate list set $\mathcal{S} = \{O_1, \ldots, O_t\}_{size=k}$, which can be either directly selected the top list according to the reward r^{sum}. Moreover, the final item lists generated from other methods (*e.g.,* DLCM and PRM) can be merged into $\mathcal{S}$ and further ranked by the successive PRank stage.

4.1.3 Efficiency Study. We make some efforts to improve the efficiency of FPSA algorithm. Specifically, we deploy the CTR and NEXT prediction model in parallel in the ranking stage to rate CTR and NEXT scores for each item, whose algorithm complexity is $O(1)$. For the algorithm in Alg. 1, we conduct the loops in lines 6-15 in parallel and adopt the min-max heaps sorting algorithm [3] to select top-k results in line 16. Totally, we reduce to the algorithm complexity if Alg. 1 to $O(n(n + k\log k))$.

Overall, the algorithm complexity of FPSA is $aO(1) + bO(n(n + k\log k))$. It is more efficient than existing list-wise methods (*e.g.,* DLCM [1] and PRM [26]) of $aO(n) + bO(n\log n)$. Here $a \gg b$ represents that the matrix calculations in deep models cost much more than the numerical calculations.

4.2 Permutation-rank (PRank) stage

As motivated, the PRank stage is proposed to provide a unified permutation-wise ranking criterion for the candidate list set generated by the PMatch stage. Most methods in current RS mainly follow the greedy strategy based on the rating scores. Though effective, such strategy neglects the permutation-variant influence in the final item list, thus is not capable to evaluate the permutations. To this end, as show in the right part of Fig. 3, we propose the LR (**L**ist **R**eward) metric to select the permutation-optimal list from the candidate list set provided by the PMatch stage, which is calculated by the rating scores of elaborately designed permutation-wise model DPWN (**D**eep **P**ermutation-**W**ise **N**etwork).

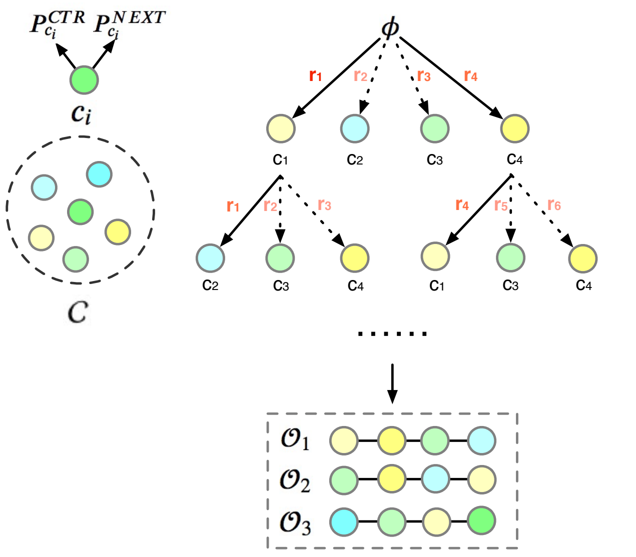

Figure 4: Illustration of the FPSA algorithm.

Algorithm 1 Fast Permutation searching algorithm (FPSA).

Input: Input ranking list C; CTR score list $\mathcal{P}^{\text{CTR}}$; NEXT score list $\mathcal{P}^{\text{NEXT}}$; Output length n; Beam size integer k; Fusion coefficient float α, β.

Output: Candidate list set $\mathcal{S}$.

```
1: New candidate list set $\mathcal{S} = \{[\phi]\}$, estimated reward set $\mathcal{R} = \{\}$;
2: // Beam search.
3: for $i = 1, 2, ..., n$ do:
4:     New candidate list set $\mathcal{S}_t = \mathcal{S}$;
5:     Clear $\mathcal{S}$ and $\mathcal{R}$;
6:     for $O \in \mathcal{S}_t$ do:
7:         for $c_i \in C$ do:
8:             if $c_i \notin O$ then:
9:                 New $O_t$ by appending $c_i$ to $O$ ;
10:                Reward $r$ = Calculate-Estimated-Reward($O_t$);
11:                $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$;
12:                $\mathcal{S} \leftarrow \mathcal{S} \cup \{O_t\}$;
13:            end if
14:        end for
15:    end for
16:    According to $\mathcal{R}$, $\mathcal{S} \leftarrow$ top $k$ of $\mathcal{S}$ ;
17: end for
18: // Calculate reward.
19: function CALCULATE-ESTIMATED-REWARD($O$)
20:     Estimated PV reward $r^{PV} = 1$, IPV reward $r^{IPV} = 0$;
21:     Transitive expose probability $p^{Expose} = 1$;
22:     for $c_i \in O$ do:
23:         $r^{PV}$ += $p^{Expose} * P_{c_i}^{\text{NEXT}}$;
24:         $r^{IPV}$ += $p^{Expose} * P_{c_i}^{\text{CTR}}$;
25:         $p^{Expose}$ *= $P_{c_i}^{\text{NEXT}}$;
26:     end for
27:     $r^{sum} = \alpha * r^{PV} + \beta * r^{IPV}$;
28:     return $r^{sum}$;
29: end function
```

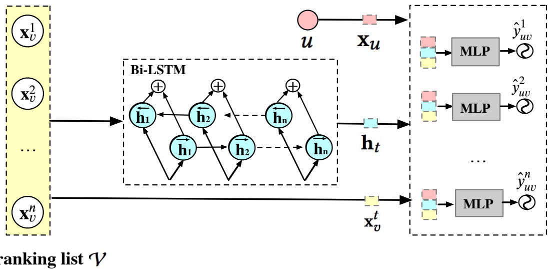

Figure 5: Technical architecture of the DPWN model.

4.2.1 Offline training. The overall architecture of DPWN is shown in Fig. 5. As motivated, DPWN is proposed to capture and model the permutation-variant influence contained by the final item list. By taking such motivation into consideration, DPWN $M(\mathbf{x}_v^t|u, \mathcal{V}; \Theta^D)$, parameterized by Θ^D, predicts the permutation-wise interaction probability between user u and the t-th item $\mathbf{x}_v^t$ in the final item list $\mathcal{V}$. Naturally, Bi-LSTM [18] is excellent at capturing such time-dependent and long-short term information within the permutation. Mathematically, the forward output state for the t-th item x_v^t can be calculated as follows:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_v^t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_v^t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{x}_v^t + \mathbf{i}_t \tanh(\mathbf{W}_{xc}\mathbf{x}_v^t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_v^t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o) \\
\overrightarrow{\mathbf{h}_t} &= \mathbf{o}_t \tanh(\mathbf{c}_t)
\end{aligned} \tag{3}$$

where $\sigma(\cdot)$ is the logistic function, and $\mathbf{i}$, $\mathbf{f}$, $\mathbf{o}$, and $\mathbf{c}$ are the input gate, forget gate, output gate and cell vectors which have the same size as $\mathbf{x}_v^t$. Shapes of weight matrices are indicated with the subscripts. Similarly, we can get the backward output state $\overleftarrow{\mathbf{h}_t}$. Then we concatenate the two output states $\mathbf{h}_t = \overrightarrow{\mathbf{h}_t} \oplus \overleftarrow{\mathbf{h}_t}$ and denote $\mathbf{h}_t$ as the sequential representation of x_v^t.

Due to the powerful ability in modeling complex interaction in the CTR prediction field [14, 27, 33, 34], we integrate the multi-layer perceptron (MLP) into DPWN for better feature interaction. Hence, we formalize the Bi-LSTM as follows:

$$M(\mathbf{x}_v^t|u, \mathcal{V}; \Theta^D) = \sigma\big(f(f(f(\mathbf{x}_u \oplus \mathbf{x}_v^t \oplus \mathbf{h}_t)))\big) \tag{4}$$

where $f(\mathbf{x}) = ReLU(\mathbf{W}\mathbf{x} + \mathbf{b})$, $\sigma(\cdot)$ is the logistic function. The parameter set of DPWN is $\Theta^D = \{\mathbf{W}_*, \mathbf{b}_*\}$, *i.e.,* the union of parameters for Bi-LSTM and MLP.

Clearly, DPWN can be optimized via binary cross-entropy loss function, which is defined as follows:

$$\mathcal{L}^D = -\frac{1}{N}\sum_{(u,\mathcal{V})\in\mathcal{R}}\sum_{x_v^t\in\mathcal{V}}\left(y_{uv}^t \log \hat{y}_{uv}^t + (1-y_{uv}^t)\log(1-\hat{y}_{uv}^t)\right) \tag{5}$$

where $\mathbb{D}$ is the training dataset. For convenience, we refer $\hat{y}_{uv}^t$ as $M(\mathbf{x}_v^t|u, \mathcal{V}; \Theta^D)$, *i.e.,* $\hat{y}_{uv}^t = M(\mathbf{x}_v^t|u, \mathcal{V}; \Theta^D)$, and $y_{uv}^t \in \{0, 1\}$ is the ground truth. We can optimize the parameters Θ^D through minimizing $\mathcal{L}^D$.

Table 2: Statistics of datasets

Description	Rec	Ad
#Users	5.48×10^7	1.06×10^6
#Items	2.08×10^7	8.27×10^5
#Records	5.08×10^8	2.04×10^6
#User-item interactions	1.44×10^8	2.04×10^7
#Avg size of C_u	147.69	100.0
#Avg size of $\mathcal{V}_u$	4.0	10.0

Note that the major difference which distinguishes our proposed DPWN from current list-wise methods [1, 25, 26] is that DPWN models the *final item list* rather than the *input ranking list.* Practically, the user is more influenced by the permutation information of the exhibited final item list.

4.2.2 Online serving. As mentioned above, we propose a unified permutation-wise LR metric based on the DPWN model, which is applied to achieve the most permutation-optimal list solution from the candidate list set generated in the PMatch stage. Specifically, we calculate the LR (list reward) score for each list O_t in the candidate list set $\mathcal{S}$ as follows:

$$LR(O_t) = \sum_{x_o^i \in O_t} \mathrm{M}(x_o^i | u, O_t) \tag{6}$$

Afterwards, the list $\mathcal{P}$ with the highest LR score is finally selected and recommended to the user, with the hope of obtaining the permutation-optimal list and most meeting the user's demands.

Finally, we refer the online serving part of PMatch and PRank stages in the PRS framework as the learned re-ranking strategy π to generate the permutation-optimal list $\mathcal{P}$ from the input ranking list C.

5 EXPERIMENTS

In this section, we perform a series of online and offline experiments to verity the effectiveness of the proposed framework PRS, with the aims of answering the following questions:

- **RQ1**: How does the proposed LR metric in the PRank stage evaluate the profits of the permutations precisely?
- **RQ2**: How does the proposed FPSA algorithm in the PMatch stage generate the candidate list set from exponential list solutions in the effective way?
- **RQ3**: How about the performance of PRS in the real-world recommender systems?

5.1 Experimental Setup

5.1.1 Datasets. We conduct extensive experiments on two real-world datasets: a proprietary dataset from Taobao application and a public dataset from Alimama, which are introduced as follows:

- **Rec** [2] dataset consists of large-scale list interaction logs collected from Taobao application, one of the most popular e-commerce platform. Besides, Rec dataset contains user profile (*e.g.,* id, age and gender), item profile (*e.g.,* id, category and brand), the input ranking list provided by the previous stages for each user and the labeled final item list.
- **Ad** [3] dataset records interactions between users and advertisements and contains user profile (*e.g.,* id, age and occupation), item profile (*e.g.,* id, campaign and brand). According to the timestamp of the user browsing the advertisement, we transform records of each user and slice them at each 10 items into list records. Moreover, we mix an additional 90 items with the final item list as the input ranking list to make it more suitable for re-ranking.

The detailed descriptions of the two datasets are shown in Tab. 2. For the both datasets, we randomly split the entire records into training and test set, *i.e.,* we utilize 90% records to predict the remaining 10% ones [4].

5.1.2 Baselines. We select two kinds of representative methods as baselines: point-wise and list-wise methods, which are widely adopted in most industrial recommender systems. Point-wise methods (*i.e.,* DNN and DeepFM) mainly predict the interaction probability for the given user-item pair by utilizing raw features derived from user and item profile. List-wise methods (*i.e.,* MIDNN, DLCM and PRM) devote to extracting list-wise information in different ways. The comparison methods are given below in detail:

- **DNN** [10] is a standard deep learning method in the industrial recommender system, which applies MLP for complex feature interaction.
- **DeepFM** [19] is a general deep model for recommendation, which combines a factorization machine component and a deep neural network component.
- **MIDNN** [35] extracts list-wise information of the input ranking list with complex handmade features engineering.
- **DLCM** [1] firstly applies GRU to encode the input ranking list into a local vector, and then combine the global vector and each feature vector to learn a powerful scoring function for list-wise re-ranking.
- **PRM** [26] applies the self-attention mechanism to explicitly capture the mutual influence between items in the input ranking list.
- **DPWN** is the elaborately designed permutation-wise model in this work, based on which we propose the LR metric.

To answer **RQ1**, we denote the sum rating scores of the recorded final item list from the baselines (*SR* for distinction), and our proposed LR metric is calculated by DPWN with the same pattern.

To answer **RQ2**, we compare the ranking result of the baselines with the list of highest estimated reward r^{sum} in FPSA under the LR metric.

To answer **RQ3**, we deploy the PRank stage with FPSA algorithm and the PMatch stage with the LR metric based on the DPWN model and report the online performance of each stage and the complete PRS framework.

Note that pair-wise and group-wise methods are not selected as baselines in our experiments due to their high training or inference complexities ($O(N^2)$) compared with point-wise ($O(1)$) or list-wise

[2]www.taobao.com

[3]https://tianchi.aliyun.com/dataset/dataDetail?dataId=56

[4]We hold out 10% training data as the validation set for parameter tuning.

Table 3: Overall performance comparison *w.r.t.* interaction probability prediction (bold: best; underline: runner-up).

Model	Rec			Ad		
	Loss	AUC	Pearson	Loss	AUC	Pearson
DNN(*SR*)	0.191	0.701	0.066	0.187	0.587	0.131
DeepFM(*SR*)	0.189	0.703	0.072	0.186	0.588	0.140
MIDNN(*SR*)	0.182	0.706	0.090	0.185	0.600	0.155
DLCM(*SR*)	0.177	0.710	0.102	0.185	0.602	0.159
PRM(*SR*)	<u>0.175</u>	<u>0.712</u>	<u>0.108</u>	<u>0.185</u>	<u>0.602</u>	<u>0.159</u>
DPWN(LR)	**0.161***	**0.730***	**0.242***	**0.183***	**0.608***	**0.172***

Table 4: Overall performance comparison *w.r.t.* the ranking results on the LR metric (bold: best; underline: runner-up).

Model	LR	RI
DNN	0.252	+13.09%
DeepFM	0.261	+9.19%
MIDNN	0.272	+4.77%
DLCM	0.274	+4.01%
PRM	<u>0.278</u>	<u>+2.51%</u>
FPSA$_{\alpha=7,\beta=1}$	**0.285***	-

models ($O(N)$), which may not be appropriate for the industrial RS.

5.1.3 Evaluation Metrics. We apply Loss (cross-entropy) and AUC (area under ROC curve) to evaluate the accuracy of model predictions. Moreover, to answer **RQ1**, we calculate the pearson correlation coefficient [4] (Pearson for short) of different metrics *w.r.t.* the ground-truth, which refers to the sum of $\mathcal{Y}^{\text{CTR}}$ (*i.e.,* list *IPV*) for each list interaction record in $\mathcal{R}$. Overall, higher Pearson score indicates the estimated list rewards of the model are more correlated to the real ones.

To illustrate the priority of our proposed FPSA algorithm to competitors for **RQ2**, we adopt the proposed LR metric to the ranking results of the FPSA algorithm and other methods. Besides, we also present the relative improvement [12, 13] (RI) w.r.t. the LR metric of the FPSA algorithm achieves over the compared methods.

For online A/B testing for **RQ3**, we choose PV and IPV metrics, which most industrial recommender systems mainly pay attention to. PV and IPV are defined as the total number of items browsed and clicked by the users, respectively.

5.1.4 Implementation. We implement all models in Tensorflow 1.4. For fair comparison, pre-training, batch normalization and regularization are not adopted in our experiments. We employ random uniform to initialize model parameters and adopt Adam as optimizer using a learning rate of 0.001. Moreover, embedding size of each feature is set to 8 and the architecture of MLP is set to [128, 64, 32]. We run each model three times and reported the mean of results. For the FPSA algorithm in Rec dataset, the beam size k is set to 50 and the output length n is set to 4.

5.1.5 Significance Test. For experimental results in Tab. 3 and 4, we use "*" to indicate that the best method is significantly different from the runner-up method based on paired t-tests at the significance level of 0.01.

5.2 Performance Comparison (RQ1)

We report the comparison results of DPWN and other baselines on two datasets in Tab. 3. The major findings from the experimental results are summarized as follows:

- Generally, list-wise methods (*i.e.,* MIDNN, DLCM and PRM) achieve more improvements in most cases than point-wise methods (*i.e.,* DNN and DeepFM). By considering the mutual influence among the input ranking list, these methods learn a refined scoring function aware of the feature distribution of the input ranking list.
- DPWN consistently yields the best performance on the Loss and AUC metrics of both datasets. In particular, DPWN improves over the best baseline PRM by 0.018, 0.006 in AUC on Rec and Ad dataset, respectively. By taking advantage of the permutation information in the final item list, DPWN shows the ability to provide more precise permutation-wise interaction probabilities of each item. Different from extracting information from the input ranking list in DLCM and PRM, Bi-LSTM helps capture the permutation-variant influence contained by the final item list, which refers as the more essential and effective factors to affect the permutation-wise predictions.
- It is worth mentioning that *LR* based on DPWN performs better on the Pearson metric than *SR* based on other competitors in both datasets. It indicates that DPWN, by leveraging the permutation-variant influence within the final list, can estimate the revenue (*i.e.,* IPV) of the permutation more accurately. Hence, its derivative LR metric is accordingly applied to ranking the candidate lists in the PRank stage.
- Compared with the experimental results on the Rec dataset, the performance lift on the Ad dataset is relatively small. One possible reason is that Ad dataset is published with randomly sampling, resulting in the inconsistent and incomplete list records and weaker intra-permutation relevance of user feedback.

5.3 FPSA Algorithm Study (RQ2)

Firstly, the rationality of using the LR metric to evaluate the model's ranking results offline is listed as follows: 1) As evidenced in the Section 5.2, our proposed LR metric based on DPWN can estimate the list revenue more accurately than the SR metric of other models; 2) It is model-based and independence of the ground-truth, which means can be directly applied to evaluate lists in the PRank stage for the online serving and ensure the online and offline consistency.

We report the comparison results of the FPSA algorithm and baselines on the Rec dataset in Tab 4 [5]. We mainly have the following three observations:

- List-wise models (*i.e.,* MIDNN, DLCM and PRM) achieve better performance than point-wise models (*i.e.,* DNN and

[5] Ad dataset is not involved since the sampling of this dataset results in the discontinuity of records.

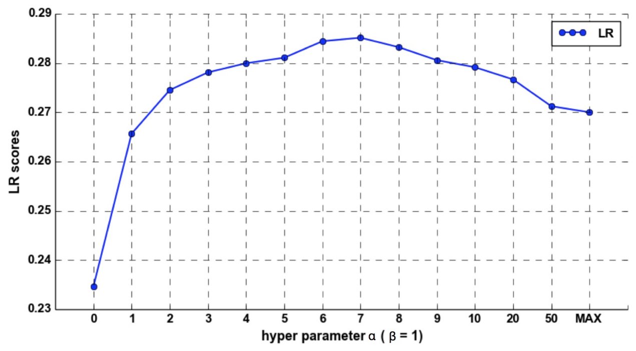

Figure 6: Impact of hyper parameters α on the LR metric.

DeepFM) on the LR metric, which indicates making the model aware of the feature distribution of the input ranking list helps achieve the better final item list.

- Our proposed FPSA algorithm with fine-tuned hyper parameters $\alpha = 7, \beta = 1$ yields the best performance on the LR metric. In particular, FPSA relatively improves over the strongest competitor PRM by 2.51%. Different from the normally applied greedy re-ranking strategy, FPSA considers the probability of continuous browsing after the item. By calculating the estimated reward at the each step of beam-searching, FPSA can dynamically balance the demands of user's browsing and interaction, so as to measure the value of the final item list more accurately.

5.3.1 Parameter Sensitivity. Practically, we have noticed the predefined hyper parameters α and β have a great influence on the final item list and the LR metric. Therefore, we conduct several grid-search experiments on these two hyper parameters to get a comprehensive understanding of the FPSA algorithm. Specifically, we fix $\beta = 1$ and tune α from 0 to MAX, and observe how LR varies under each hyper parameter. Note that setting α to 0 or MAX means that the final item list is in a descending order *w.r.t.* the NEXT or CTR score, respectively.

As shown in Fig. 6, we present the influence of hyper parameters on the final item lists. We observe the LR metric *w.r.t.* α declines from 7 to 0 and from 7 to MAX, sharply at 0. It indicates that excessively guiding users to browse or interact may miss his/her demands or increase his/her fatigue, respectively. When setting α to 7, recommendation results of FPSA take a browsing and interacting balance and deliver the best user experience.

5.4 Online A/B Testing (RQ3)

To verify the effectiveness of our proposed framework PRS in the real-world settings, PRS has been fully deployed in the *Minidetail* scenario, one of the most popular recommendation scenario of Taobao application. AS shown in the left part of Fig. 7, in this waterfall scenario, users can browse and interact with items sequentially and endlessly. The fixed-length recommended results are displayed to the user when he/she reaches the bottom of the current page. Considering the potential problem of high latency to the user experience, it brings great challenges to deploy PRS into production.

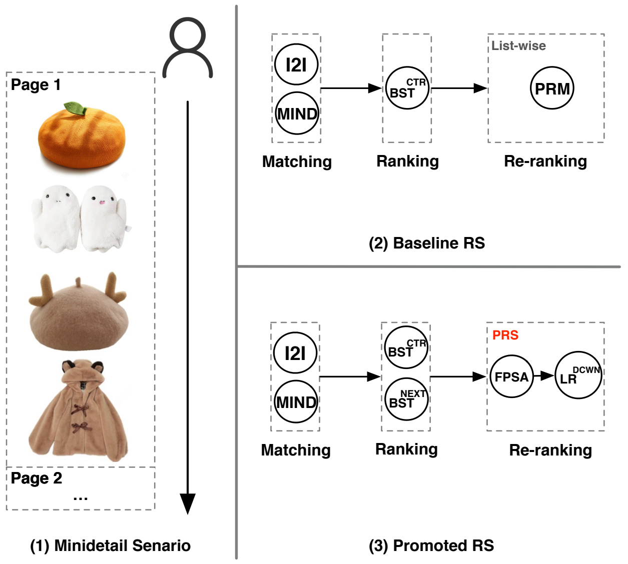

Figure 7: (1) is the *Minidetail* scenario. (2) and (3) present the difference of the baseline RS and the promoted RS with the PRS framework.

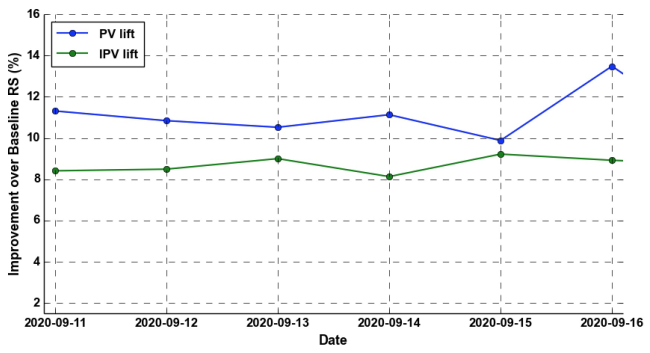

Figure 8: Online improvements on PV and IPV metrics in a week.

We first introduce our effective baseline RS, and then the efforts of merging the PRS framework into the promoted RS. AS shown in the right part of Fig. 7, the baseline RS consists of the matching, ranking and re-ranking stages, equipped with I2I [31] and MIND [22] in matching, BST [8] in ranking and PRM [26] in re-ranking. For the FPSA algorithm in the PMatch stage, we concurrently deploy BST^{CTR} and BST^{NEXT} to calculate the CTR and NEXT scores for each item in the ranking stage, without extra time cost. Then multiple candidate final item lists are generated by the FPSA algorithm, and then ranked by the LR metric calculated by the deployed DPWN model in the PRank stage. Finally, the final item list with highest LR scores are recommend to the user.

Thanks to the above efforts, we successfully deployed PRS into production and report the online relative improvements from "2020-09-11" to "2020-09-17" in the Fig. 8. Specifically, the FPSA algorithm in the PMatch stage obtains **11.0**% lift on PV metric and **2.6**% lift on IPV metric, with the average cost of **1.2** milliseconds for online inference. Afterwards, the LR metric based on the DPWN model in the PRank stage obtains **6.1**% lift on IPV metric, with the average cost of **6.2** milliseconds for online inference. Totally, the PRS framework takes only **7.3** milliseconds to obtain the significant **11.0**% lift on PV metric and **8.7**% improvement on IPV metric. Considering the massive influenced users and maturity of the baseline RS, the promotion of recommendation performance verifies the effectiveness of our proposed framework PRS.

6 CONCLUSION

In this paper, we highlight the importance of the permutation knowledge in the final ranking list and address how to leverage and transform such information for better recommendation. We propose a novel two-stage permutation-wise framework PRS in the re-ranking stage, consisting of the PMatch and PRank stages successively. The PMatch stage with the proposed FPSA is designed to generate multiple candidate item lists, and the PRank stage with the LR metric based on the DPWN model provide a uniform ranking criterion to select the most effective list for the user. Extensive experiments on both industrial and benchmark datasets demonstrate the effectiveness of our framework compared to state-of-the-art point-wise and list-wise methods. Moreover, PRS has also achieved impressive improvements on the PV and IPV metrics in online experiments after successful deployment in one popular recommendation scenario of Taobao application.

In the future, we will concentrate on proposing more effective methods in the PMatch and PRank stages. Besides, with the improvement of computing efficiency, PRS has the potential to replace matching and ranking stages with PMatch and PRank stages, respectively.

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In *SIGIR*. 135–144.

[2] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning groupwise multivariate scoring functions using deep neural networks. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 85–92.

[3] Michael D Atkinson, J-R Sack, Nicola Santoro, and Thomas Strothotte. 1986. Min-max heaps and generalized priority queues. *Commun. ACM* 29, 10 (1986), 996–1000.

[4] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*. Springer, 1–4.

[5] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *ICML*. 89–96.

[6] Christopher JC Burges. [n. d.]. From ranknet to lambdarank to lambdamart: An overview. *Learning* ([n. d.]).

[7] Laming Chen, Guoxin Zhang, and Eric Zhou. 2018. Fast greedy map inference for determinantal point process to improve recommendation diversity. In *NeurIPS*. 5622–5633.

[8] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.

[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 *RecSys Workshop*. 7–10.

[10] Covington, Paul, Adams, Jay, Sargin, and Emre. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.

[11] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *WWW*. 271–280.

[12] Yufei Feng, Binbin Hu, Fuyu Lv, Qingwen Liu, Zhiqiang Zhang, and Wenwu Ou. 2020. ATBRG: Adaptive Target-Behavior Relational Graph Network for Effective Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2231–2240.

[13] Yufei Feng, Fuyu Lv, Binbin Hu, Fei Sun, Kun Kuang, Yang Liu, Qingwen Liu, and Wenwu Ou. 2020. MTBRN: Multiplex Target-Behavior Relation Enhanced Network for Click-Through Rate Prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2421–2428.

[14] 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 *IJCAI*. 2301–2307.

[15] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. 2019. Deepmdp: Learning continuous latent space models for representation learning. *arXiv preprint arXiv:1906.02736* (2019).

[16] Anupriya Gogna and Angshul Majumdar. 2017. Balancing accuracy and diversity in recommendations using matrix completion framework. *Knowledge-Based Systems* 125 (2017), 83–95.

[17] Carlos A Gomez-Uribe and Neil Hunt. 2015. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems* 6, 4 (2015), 1–19.

[18] Alex Graves and Jrgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18, 5-6 (2005), 602–610.

[19] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*. 1725–1731.

[20] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *SIGKDD*. 217–226.

[21] Roger Levy and T Florian Jaeger. 2007. Speakers optimize information density through syntactic reduction. *Advances in neural information processing systems* 19 (2007), 849.

[22] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2615–2623.

[23] Lei Li, Dingding Wang, Tao Li, Daniel Knox, and Balaji Padmanabhan. 2011. SCENE: A Scalable Two-Stage Personalized News Recommendation System *(SIGIR '11)*. Association for Computing Machinery, New York, NY, USA, 125–134. https://doi.org/10.1145/2009916.2009937

[24] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *ACM IUI*. 31–40.

[25] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. Setrank: Learning a permutation-invariant ranking model for information retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 499–508.

[26] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, and Dan Pei. 2019. Personalized Re-Ranking for Recommendation. In *RecSys*. 3–11.

[27] 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 *SIGKDD*. 2671–2679.

[28] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[29] Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H Chi, and Jennifer Gillenwater. 2018. Practical diversified recommendations on youtube with determinantal point processes. In *CIKM*. 2165–2173.

[30] Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* (2016).

[31] Xiaoyong Yang, Yadong Zhu, Yi Zhang, Xiaobo Wang, and Quan Yuan. 2020. Large Scale Product Graph Construction for Recommendation in E-commerce. *arXiv preprint arXiv:2010.05525* (2020).

[32] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR*. 287–294.

[33] 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 *AAAI*. 5941–5948.

[34] 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 *SIGKDD*. 1059–1068.

[35] Tao Zhuang, Wenwu Ou, and Zhirong Wang. 2018. Globally Optimized Mutual Influence Aware Ranking in E-Commerce Search. In *IJCAI*. 3725–3731.