GRN: Generative Rerank Network for Context-wise Recommendation

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

ABSTRACT

Reranking is attracting incremental attention in the recommender systems, which rearranges the input ranking list into the final ranking list to better meet user demands. Most existing methods greedily rerank candidates through the rating scores from point-wise or list-wise models. Despite effectiveness, neglecting the mutual influence between each item and its contexts in the final ranking list often makes the greedy strategy based reranking methods sub-optimal. In this work, we propose a new context-wise reranking framework named **G**enerative **R**erank **N**etwork (GRN for short). Specifically, we first design the evaluator, which applies Bi-LSTM and self-attention mechanism to model the contextual information in the labeled final ranking list and predict the interaction probability of each item more precisely. Afterwards, we elaborate on the generator, equipped with GRU, attention mechanism and pointer network to select the item from the input ranking list step by step. Finally, we apply cross-entropy loss to train the evaluator and, subsequently, policy gradient to optimize the generator under the guidance of the evaluator. Empirical results show that GRN consistently and significantly outperforms state-of-the-art point-wise and list-wise methods. Moreover, GRN has achieved a performance improvement of 5.2% on PV and 6.1% on IPV metric after the successful deployment in one popular recommendation scenario of Taobao application.

ACM Reference Format:
Yufei Feng, Binbin Hu, Yu Gong, Fei Sun, Qingwen Liu, Wenwu Ou. 2021. GRN: Generative Rerank Network for Context-wise Recommendation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Recommender systems (RS) have been widely deployed in various web-scale applications, including e-commerce [9, 11, 24, 32], social media [9, 14] and online news [10, 19, 20]. Owing to the great impact on user satisfaction as well as the revenue of the RS, recent attention is increasingly shirting towards the reranking stage, which is typically designed to rearrange the input ranking list generated by previous stages (i.e., matching and ranking). With the rapid development of deep learning techniques, various reranking algorithms have been proposed to address the interior relevance in the input ranking list for improving recommendation performance.

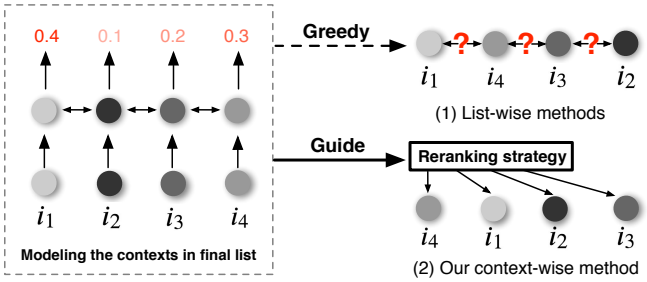

Figure 1: The difference of existing works and ours. Greedily reranking changes the original contexts and lead to inaccurate prediction in the new permutation. Our work attempts to learn a context-wise reranking strategy under the guidance of the well-trained context-wise model.

In particular, several recent efforts have attempted to follow the greedy strategy for reranking on items with refined rating scores, which mainly fall into two groups, global point-wise models [8, 9, 16] and local list-wise models [1, 4, 6, 22, 33]. Global point-wise models learn a ranking function with the labeled user-item pairs. In spite of effectiveness, they ignore different feature distributions within the input ranking list generated for each user. In fact, users may show different concerns about different input ranking lists, such as the price of the snacks and the brand of the electronics. To solve this, several local list-wise models have been proposed to refine the ranking scores by taking feature distributions of the input ranking list into account. Unfortunately, these methods commonly follow the greedy strategy for item rearrangements, which changes the contexts of each item from the initial list to the final list, resulting in imprecise estimation under the new item permutation. As shown in Fig. 1, the rating scores predicted by modeling the contexts of items (i_1, i_2, i_3, i_4) in the final list could be $(0.4, 0.1, 0.2, 0.3)$. Specifically, 0.3 represents the contextual interaction probability when placed before i_2 and after i_1, i_4. Once the candidates are rearranged according to the ratings scores to (i_1, i_4, i_3, i_2), the contextual items of i_3 are modified, leading to inaccurate estimation in the new permutation and sub-optimal item arrangements.

How to better leverage contexts in the reranking stage remains a great challenge in RS. Intuitively, instead of directly recommending a cheap item, placing a more expensive item ahead can stimulate the user's desire to buy the cheaper item. Besides, placing all the items of the user's most interest in the front may reduce the possibility of

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

his/her continue browsing. Such generalized contextual knowledge is of crucial importance in the item rearrangement. Unfortunately, such contexts are unable to determine and capture since the final ranking list is not given. Inspired by continuous advances in actor-critic reinforcement learning [21, 25, 26], one practical and effective solution is to learn a context-wise reranking strategy under the guidance of the well-trained context-wise model. Specifically, as illustrated in Fig. 1, we transform the context-wise reranking objective into the following two tasks: (1) The local context-wise model is designed to predict the contextual interaction probability (*e.g.,* click-through rate and conversion rate) more precisely, by fully leveraging the contexts of each item from the labeled records between users and item lists; (2) The reranking strategy is meant to obtain the rearranged item list from the input ranking list. Moreover, the reranking strategy is converged to be context-wisely optimal under the guidance of the local context-wise model. In this way, by considering the contextual information, the reranking strategy can reach the context-wise item arrangements by distilling contextual knowledge from the local context-wise model, and then bring better experience for users.

In this work, with the above discussions in mind, we propose a novel context-wise framework named **G**enerative **R**erank **N**etwork (GRN for short), which is composed of three parts: (1) Evaluator. To better refine the contextual interaction probability of each item in the final ranking list, we employ Bi-LSTM and self-attention mechanism to capture sequential information alongside the list and mutual influence between items, respectively; (2) Generator. With the aim of learning the reranking strategy to generate the final ranking list, we apply pointer network combined with GRU and attention mechanism to select appropriate item from the input ranking list at each step; (3) Training procedure. We adopt cross-entropy loss to train evaluator and, subsequently, policy gradient with technically designed advantage reward to train the generator.

In sum, we make the following contributions:

- We highlight the necessity of modeling the contexts in the final item list to each item for more precise prediction and better item rearrangements. We propose a practical and effective solution is to learn a context-wise reranking strategy under the guidance of the well-trained context-wise model.
- We propose a novel context-wise reranking framework named GRN, which simultaneous employs the well-designed evaluator and generator in an cooperative manner, with the aims of comprehensively capturing evolving intent and mutual influence implied in input and final ranking list.
- We perform a series of experiments on a public dataset from Alimama and a proprietary dataset from Taobao application. Online and offline experimental results demonstrate that GRN consistently and significantly outperforms various state-of-the-art methods.

The remainder of this paper is organized as follows. Section 2 discusses related work for reranking, including point-wise, pair-wise and list-wise models. Section 3 presents some preliminary knowledge as well as the task description. Then, we propose the generative rerank network in Section 4. Experiments and detailed analysis are reported in Section 5. Finally, we conclude the paper in Section 6.

2 RELATED WORK

Reranking usually serves as the last stage after matching and ranking stages in one typical recommender system. In this section, we review the most related studies in the reranking stage of recommender systems, where an input ranking list provided by previous stages is rearranged into the final ranking list and expected to better meet user demands. Most reranking methods can be broadly classified into three categories: point-wise, pair-wise and list-wise methods.

- **Point-wise** methods [8, 9, 16] 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.
- **Pair-wise** models work by considering the semantic distance of an item pair every time. Following this line, a surge of works are proposed to techinically designed pair-wise loss functions to compare any item pair in the input ranking list with well-designed architecture, including RankSVM [18] based on SVM, GBRank [30] based on GBDT as well as RankNet [5] and DSF [2] based on emerging deep neural networks. 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 [33] works by handmade global list-wise features, while it requires much domain knowledge and decreases its generalization performance. DLCM [1] and PRM [22] apply GRU and transformer to encode the list-wise information of the input ranking list for better prediction, respectively. Though effective, this type of list-wise models do not escape the paradigm of greedy ranking based on predicted scores, where the adjustment of the final ranking list greatly changes the contextual information of each item.

Most closest to our work are MIRNN [33] and Seq2slate [4], which directly learns a step-greedy reranking strategy to generate the final ranking list through RNN and pointer network, respectively. We argue that only considering the preceding information is insufficient and incomplete for the optimal item rearrangements since the interaction probability of each item is heavily affected by both preceding and subsequent ones. In this work, we upgrade the step-greedy strategy based reranking methods to the context-wise reranking strategy.

There are also some works [7, 12, 13, 29] focusing on making the trade-off between relevance and diversity in the reranking stage. Different from these works, GRN is an end-to-end context-wise reranking framework, which may automatically generate diverse or alike recommendaion results for the only sake of effectiveness. Besides, group-wise methods [2] determine the priority of items by multiple documents in the list. Though effective, the computation complexity of group-wise methods is at least $O(N^2)$, which may not be appropriate for industrial recommender systems.

3 PRELIMINARY

In general, a mature recommender system (*e.g.,* e-commerce and news) is composed of three stages chronologically: matching, ranking and reranking. In this paper, we focus on the final reranking stage, whose input is the ranking list produced by the previous two stage (*i.e.,* matching and ranking). The goal of the reranking is to elaborately select candidates from the input ranking list and rearrange them into the final ranking list, followed by the exhibition for users. Mathematically, with user set $\mathcal{U}$ and item set $\mathcal{I}$, we denote list interaction records as $\mathcal{R} = \{(u, C, \mathcal{V}, \mathcal{Y} | u \in \mathcal{U}, \mathcal{V} \subset C \subset \mathcal{I}\}$. Here, C and $\mathcal{V}$ represent the recorded input ranking list with m items for reranking stage and the recorded final ranking list with n items exhibited to user u, respectively. Intuitively, $n \leq m$. $y^t_{uv} \in \mathcal{Y}$ is the implicit feedback of user u *w.r.t.* t-th item $v_t \in \mathcal{V}$, which can be defined as follows: $y^t_{uv} = 1$ when interaction (*e.g.,* click) is observed, and $y^t_{uv} = 0$ otherwise. In the real-world industrial recommender systems, each user u is associated with a user profile x_u consisting of sparse features x^u_s (*e.g.,* user id and gender) and dense features x^u_d (*e.g.,* age), while each item i is also associated with a item profile x_i consisting of sparse features x^s_i (*e.g.,* item id and brand) and dense features x^d_i (*e.g.,* price) .

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

Definition 1. ***Task Description****. Given a certain user u, involving his/her input ranking list C, the goal is to learn a reranking strategy $\pi : C \xrightarrow{\pi} O$, which aims to select and rearrange items from C, and subsequently recommend a final ranking list O that are of the most interest to u.*

Many efforts have been made for reranking task, while most of these works ignore the contextual information in the final ranking list (i.e., $\mathcal{V}$), resulting in sub-optimal performance. In practice, such contextual knowledge is of crucial importance, since users are commonly sensitive to the permutation of items during browsing.

4 THE PROPOSED FRAMEWORK

In this section, we introduce our proposed framework GRN, which aims to achieve context-wise item arrangements in the reranking stage of recommender systems. Overall, GRN is composed of three parts: (1) To predict the contextual probability of being interacted in the labeled final ranking list $\mathcal{V}$, we propose the local context-wise model named evaluator, where Bi-LSTM and self-attention mechanism are applied to capture the sequential information and mutual influence between items; (2) To generate the final ranking list O from the input ranking list C, we elaborate on the model design of generator. Specifically, we equip the generator with pointer network, GRU and attention mechanism, with the hope of the abilities to select the most suitable item from C and capture adjacent information; (3) To converge the generator to a preeminent reranking strategy under the guidance of evaluator, we pay special attention to designing the training procedure. First, we adopt cross-entropy loss to train the evaluator with labeled list records. Afterwards, policy gradient cooperated with proposed advantage reward are applied to train the generator. The key notations we will use throughout the article are summarized in Table 1.

Table 1: Key notations.

Notations	Description
$\mathcal{U}, \mathcal{I}$	the set of users and items, respectively
$C, \mathcal{V}$	the recorded input and final ranking list, respectively
$\mathcal{Y}$	the implicit feedback towards the ranking list
m, n	the number of items in the recorded input and final ranking list, respectively
π	the learned reranking strategy
O	the generated final ranking list
x^u_s, x^u_d	the sparse and dense feature set for users, respectively
x^i_s, x^i_s	the sparse and dense feature set for items, respectively
Θ^G, Θ^E	parameters of generator and evaluator, respectively

First of all, we begin with the representations of users and items, which are basic inputs of our proposed model. Following previous works [9, 32], we parameterize[1] the available profiles into vector representations for users and items. Given a user u, associated with sparse features x^s_u and dense features x^d_u, we embed each sparse feature value into d-dimensional space, while handle dense features standardization or batch normalization to ensure normal distribution. Subsequently, each user u can be represented as $\mathbf{x}_u \in \mathbb{R}^{|x^s_u| \times d + |x^d_u|}$, where $|x^s_u|$ and $|x^d_u|$ denote the size of sparse and dense feature space of user u, respectively. Similarly, we represent each item i as $\mathbf{x}_i \in \mathbb{R}^{|x^s_i| \times d + |x^d_i|}$. Naturally, we represent the input ranking list C as $\mathbf{C} = [\mathbf{x}^1_c, ..., \mathbf{x}^m_c]$ and the final ranking list $\mathcal{V}$ as $\mathbf{V} = [\mathbf{x}^1_v, ..., \mathbf{x}^n_v]$, where m and n is the number of items in the input ranking list and final ranking list, respectively.

In the ensuing sections, we shall zoom into the proposed evaluator and generator for GRN.

4.1 Evaluator

As motivated, in the context-wise view, the underlying reasons driving a user to interact with a item in a list may depend on following two aspects: (1) the two-way evolution of user's intent during browsing; (2) the mutual influence between items in the item list. Thus, by taking above factors into consideration, our evaluator $E(\mathbf{x}^t_v | u, \mathcal{V}; \Theta^E)$, parameterized by Θ^E, estimates the contextual interaction probability between user u and the t-th item $\mathbf{x}^t_v$ in the final ranking list $\mathcal{V}$. Specifically, our evaluator aims to capture following two aspects of information implicated in sequences.

- **Intent evolution.** Naturally, user intent keeps bi-directionally evolving when browsing items sequentially, and thus, it is imperative to capture such dynamics of intent for modeling user preference. Here, we adopt the commonly used Bi-LSTM [17]

[1]Note that the parameters are not shared in the evaluator and generator.

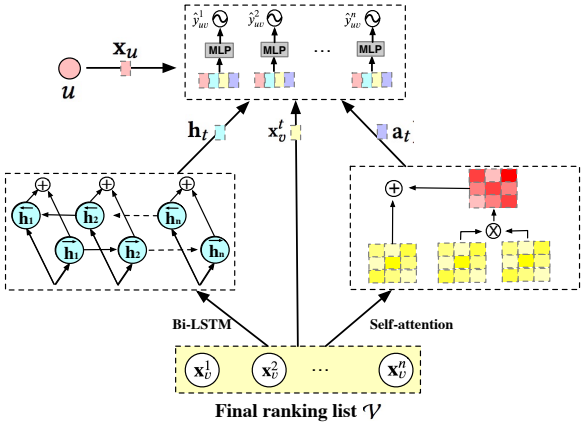

Figure 2: The overall architecture of evaluator in GRN.

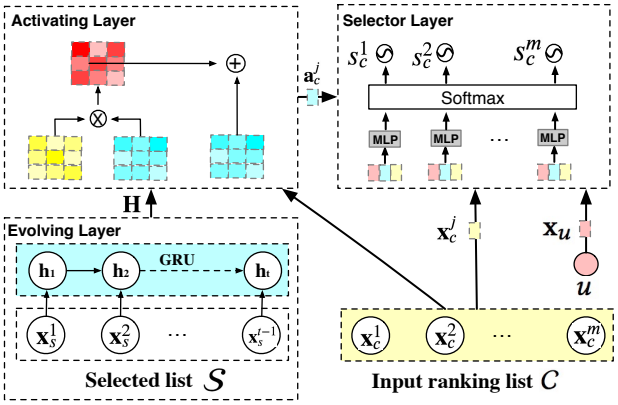

Figure 3: The overall architecture of generator in GRN.

to deal with sequences for long-term and short-term preference. Mathematically, the forward output state for the t-th item x_v^t can be calculated as follows:

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

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.

- **Mutual influence.** That is, any two items can affect each other, although they are far away from each other in the final ranking list. To achieve this goal, we apply the recently emerging self-attention mechanism [27] to directly model the mutual influences for any two items regardless the distances between them. Formally, we implement it as follows:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{V}\mathbf{V}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{2}$$

where d_k is the dimension of each item representation in $\mathcal{V}$. Obviously, we can easily extend it to multi-head attention for the stable training process. We denote $\mathbf{a}_t$, the t-th representation in $\mathbf{A}$, as the mutual representation of x_v^t.

Due to the powerful ability in modeling complex interaction in the CTR prediction field [11, 23, 31, 32], we integrate the multi-layer perceptron (MLP) into the evaluator for better feature interaction. Hence, we formalize our evaluator as follows:

$$E(\mathbf{x}_v^t|u,\mathcal{V};\Theta^E) = \sigma\big(f(f(f(\mathbf{x}_u \oplus \mathbf{x}_v^t \oplus \mathbf{h}_t \oplus \mathbf{a}_t)))\big) \tag{3}$$

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

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

$$\mathcal{L}^E = -\frac{1}{N}\sum_{(u,\mathcal{V})\in\mathcal{R}}\sum_{x_v^t\in\mathcal{V}}\big(y_{uv}^t \log\hat{y}_{uv}^t + (1-y_{uv}^t)\log(1-\hat{y}_{uv}^t)\big) \tag{4}$$

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

4.2 Generator

The goal of our generator $G(u, C; \Theta^G)$, parameterized by Θ^G, is to learn a reranking strategy for item selection from input ranking list C, that are of the most interest to user u. In the following sections, we focus on how to select the appropriate item at each step by considering the user's dynamic state during browsing, besides how to generate the final ranking list and train the generator.

Evolving Layer. As motivated, user's intent or interest will change over time, which derive us to learn dynamic representation for the target user to adapt for the evolution of the browsing history. For this purpose, we aim to distill the sequential information into the state representation at each step for the target user u, which achieved by the *time-efficient* GRU module. Formally, at the t-th step, let $\mathcal{S} = [\mathbf{x}_s^0, \mathbf{x}_s^1, ..., \mathbf{x}_s^{t-1}]$ be the $t-1$ items selected from C [2], the output of GRU module can be calculated as follows,

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_z\mathbf{x}_l^{t-1} + \mathbf{U}_z\mathbf{h}_{t-1}) \\
\mathbf{r}_t &= \sigma(\mathbf{W}_t\mathbf{x}_l^{t-1} + \mathbf{U}_t\mathbf{h}_{t-1}) \\
\widetilde{\mathbf{h}}_t &= \tanh(\mathbf{W}\mathbf{x}_l^{t-1} + \mathbf{U}(\mathbf{r}_t\mathbf{h}_{t-1})) \\
\mathbf{h}_t &= (1-\mathbf{z}_t)\mathbf{h}_{t-1} + \mathbf{z}_t\widetilde{\mathbf{h}}_t
\end{aligned} \tag{5}$$

where $\mathbf{W}_z, \mathbf{U}_z, \mathbf{W}_t, \mathbf{U}_t, \mathbf{W}$ and $\mathbf{U}$ are trainable variables. We denote $\mathbf{H} = [\mathbf{h}_1, ..., \mathbf{h}_t]$ as the sequential encoding of the selected list $\mathcal{S}$.

[2] Specially, for the first step, $\mathbf{x}_l^0$ is a trainable vector which has the same embedding size with $\mathbf{x}_l^t$

Activating Layer. Typically, the selected items have different influence towards the remaining items in the input ranking list. It is a common phenomenon that a user may get tired of the clothing category after browsing abundant different clothes. Hence, in this case, recommending other categories matching his/her interest could be a better choice. Inspired by vanilla attention mechanism [3] , we learn the attention weights over remaining items in the input ranking list conditioned on the sequential representation of selected items. Given the representation of j-th remaining item $\mathbf{x}_c^j$ and the sequential representations of the selected list $\mathbf{H}$, we adopt weighted sum to generate the new representation for the j-th in the input ranking list with the corresponding attention weights as follows:

$$\begin{aligned} a_h^i &= \frac{\exp(\mathbf{h}_i \mathbf{W} \mathbf{x}_c^j)}{\sum_{i=1}^t \exp(\mathbf{h}_i \mathbf{W} \mathbf{x}_c^j)} \\ \mathbf{a}_c^j &= \sum_{i=1}^t a_h^i \mathbf{h}_i \end{aligned} \tag{6}$$

Selector Layer. After obtaining the representations for items (*i.e.,* $\mathbf{a}_c^j$) in input ranking list, we now aim to select the most suitable item, which best matches the potential interest for the target user at the t-th step. Here, we choose pointer network [28] to achieve this goal. Formally, for the j-th item $\mathbf{x}_c^j$ in the input ranking list $\mathbf{C}$, we first apply MLP for better feature interaction at the t-th step:

$$\widetilde{s}_c^j = f(f(f(\mathbf{x}_u \oplus \mathbf{x}_c^j \oplus \mathbf{a}_c^j))) \tag{7}$$

Afterwards, we apply the softmax function to normalize the vector $\widetilde{s}_c^j$ as follows:

$$s_c^j = \frac{\exp(\widetilde{s}_c^j)}{\sum_j^m \exp(\widetilde{s}_c^j)} \tag{8}$$

where m is the number of items in $\mathbf{C}$. Then, at t-th step, our generator will select the item with the highest selection probability excluding the selected items, which can be formalized as follows:

$$G^t(u, C; \mathbf{\Theta}^G) = x_c^{\arg\max_j s_c^j} \quad s.t.\ x_c^{\arg\max_j s_c^j} \notin \mathcal{S} \tag{9}$$

The parameter set of the generator is $\mathbf{\Theta}^G = \{\mathbf{U}_*, \mathbf{W}_*, \mathbf{b}_*\}$, *i.e.,* the union of parameters for GRU module and point network.

List Generation. Our generator is recurrently repeated until the generated final ranking list reaches the pre-defined length (*i.e.,* n). After repeating calculating $G^t(u, C; \mathbf{\Theta}^G)$ for n times, we get the reranking strategy $\pi = [G^1(u, C; \mathbf{\Theta}^G), ..., G^n(u, C; \mathbf{\Theta}^G)]$ and the generated final ranking list $O = [x_o^1, ..., x_o^n]$, correspondingly.

Advantage Reward. As illustrated earlier, we utilize the evaluator to guide the optimization of the generator for the context-wise reranking strategy through policy gradient. As mentioned above, at each step, our generator will select the most suitable item from C in the light of the probability distribution in Eq. 8. Besides, we propose the *advantage reward* to estimate the actual reward of each item in the final ranking list more comprehensively. Specifically, the advantage reward consists of two parts:

- **Self reward**: As motivated, the interaction probability of itself in the final ranking list heavily affected by its context. Here we use the predicted contextual score by the evaluator to denote the its own reward in the list, which is calculated as follows:

$$r^{self}(x_o^t | u, O) = \mathrm{E}(x_o^t | u, O; \mathbf{\Theta}^E) \tag{10}$$

- **Differential reward**: Comprehensively, besides the self reward, each item brings differences in the interaction probability of other items in the list. For example, though selecting an unattractive item is generally not a good idea, it is appreciated if it can increase the interaction probability of other items. Hence, we propose the differential reward, with the assistance of the generator and calculated as follows:

$$\begin{aligned} r^{diff}(x_o^t | u, O) = &\sum_{x_o^i \in O^-} \mathrm{E}(x_o^i | u, O; \mathbf{\Theta}^E) - \\ &\sum_{x_o^i \in O^-} \mathrm{E}(x_o^i | u, O^-; \mathbf{\Theta}^E) \end{aligned} \tag{11}$$

where O^- is the item list which removes x_o^t from O.

By combining both above rewards, we define the advantage reward of t-th item x_o^t in O as follows:

$$\begin{aligned} &r(x_o^t | u, O) = r^{self}(x_o^t | u, O) + r^{diff}(x_o^t | u, O) \\ &= (\mathrm{E}(x_o^t | u, O; \mathbf{\Theta}^E) + \sum_{x_o^i \in O^-} \mathrm{E}(x_o^i | u, O; \mathbf{\Theta}^E)) - \sum_{x_o^i \in O^-} \mathrm{E}(x_o^i | u, O^-; \mathbf{\Theta}^E) \\ &= \sum_{x_o^i \in O} \mathrm{E}(x_o^i | u, O; \mathbf{\Theta}^E) - \sum_{x_o^i \in O^-} \mathrm{E}(x_o^i | u, O^-; \mathbf{\Theta}^E) \end{aligned} \tag{12}$$

Finally, the loss function of the generator is defined as follows:

$$\mathcal{L}^G = -\frac{1}{N} \sum_{(u,C) \in \mathcal{R}} \sum_{x_o^t \in O} r(x_o^t | u, O) \log s_c^j. \tag{13}$$

Here s_c^j represents the sampling probability of x_o^t from C in Eq. 8. We can optimize the parameters $\mathbf{\Theta}^G$ through minimizing $\mathcal{L}^G$.

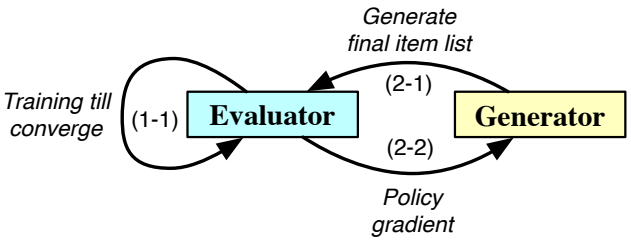

Figure 4: The illustration of the training procedure for GRN.

4.3 Training Procedure

we adopt the two-stage optimization strategy to train GRN, which is illustrated in Fig. 4. We first utilize the list interaction records to optimize $\mathbf{\Theta}^E$ and thus improve the evaluator until converge ((1-1) in Fig. 4). Next, we fix $\mathbf{\Theta}^E$, and optimize $\mathbf{\Theta}^G$ in order to produce the final ranking list and better meet user's demands. Specifically, the generator will produce the final ranking list for the evaluator((2-1) in Fig. 4), and then update the paratemer $\mathbf{\Theta}^G$ via policy gradient with advantage reward based on the evaluator((2-2) in Fig. 4). The above process is repeated for more iterations until our generator

Algorithm 1 Training procedure for GRN

Input: List interaction records as $\mathcal{R} = \{(u, \mathcal{V}, C, \mathcal{Y})\}$; Evaluator $E(\mathbf{x}_o^t | u, \mathcal{V}; \Theta^E)$; Generator $G(u, C; \Theta^G)$
Output: Converged parameters Θ^E and Θ^G;

// **Evaluator training**
while Θ^E not converge **do**
  Calculate prediction scores of evaluator by Eq. 1 ~ 3;
  Optimizing Θ^E with loss in Eq. 4
end while
// **Generator training**
while Θ^G not converge **do**
  New final ranking list O;
  for $t = 1, 2, ..., n$ **do**:
    Generate the t-th item x_o^t from C by Eq. 5 ~ 9;
    $O \leftarrow O \cup x_o^t$;
  end for
  Calculate the advantage reward of each item in O by Eq. 10 ~ 12;
  Optimize Θ^G with loss in Eq. 13
end while

Table 2: Statistics of datasets

Description	Rec	Ad
#Users	2.16×10^8	1.06×10^6
#Items	4.36×10^7	8.27×10^5
#Records	3.01×10^9	2.04×10^6
#User-item interactions	7.07×10^8	2.04×10^7
#Avg size of C_u	83.86	100.0
#Avg size of $\mathcal{V}_u$	4.26	10.0

converges. Overall, the training procedure for GRN is outlined in Algorithm 1.

5 EXPERIMENTS

In this section, we perform a series of experiments on two real-world datasets, with the aims of answering the following research questions:

- **RQ1**: How does GRN predict the interaction probability more precisely and converge to a better reranking strategy compared with SOTA models on the reranking task?
- **RQ2**: How do the well-designed components of GRN (*e.g.,* Bi-LSTM, self-attention, etc.) influence the performance of GRN?
- **RQ3**: How does the generator of GRN provide a better reranking strategy intuitively?
- **RQ4**: How about the performance of the generator of GRN in the real-world recommendation scenarios with efficient deployment?

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** [3] dataset consists of large-scale list interaction logs collected from Taobao application, one of the most popular e-commerce platform in China. 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 ranking list.
- **Ad** [4] 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 ranking list as the input ranking list to make it more suitable for reranking.

The detailed descriptions of the two datasets are shown in Table 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% interactions [5].

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 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 , PRM and Seq2Slate) devote to extracting list-wise information with different well-designed principles. The comparison methods are given below in detail:

- **DNN** [9] is a standard deep learning method in the industrial recommender system, which applies MLP for complex feature interaction.
- **DeepFM** [16] is a general deep model for recommendation, which combines a factorization machine component and a deep neural network component.
- **MIDNN** [33] 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 reranking.
- **PRM** [22] applies the self-attention mechanism to explicitly capture the mutual influence between items in the input ranking list.
- **Seq2Slate** [4] applies rnn and pointer network to encode the previous selected items and select the most appropriate item at each step.
- **GRN** is the novel context-wise framework proposed in this paper, which consists of the evaluator (*i.e.,* $\mathbf{GRN}_{\mathcal{E}}$) for predicting interaction probabilities and the generator (*i.e.,* $\mathbf{GRN}_{\mathcal{G}}$) for generating reranking results.

[3] https://www.taobao.com
[4] https://tianchi.aliyun.com/dataset/dataDetail?dataId=56
[5] We hold out 10% training data as the validation set for parameter tuning.

It is worthwhile to note that pair-wise and group-wise methods are not selected as baselines in our experiments due to their high training or inference complexities (at least $O(N^2)$) compared with point-wise ($O(1)$) or list-wise and context-wise ($O(N)$) models. Moreover, these methods usually achieve relatively worse performance, which have been reported in many previous studies [1, 6, 22].

5.1.3 Evaluation Metrics. To answer **RQ1**, we adopt different criteria to evaluate the model performance in following two aspects: (1) To evaluate the accuracy of model predictions for $\text{GRN}_{\mathcal{E}}$ and baselines, Loss (cross-entropy), AUC (area under ROC curve) and GAUC [32] (average intra-user AUC) are applied. (2) To evaluate the reranking results of $\text{GRN}_{\mathcal{G}}$ and baselines, apart from NDCG (normalized discounted cumulative gain) metrics, we design the LR (list reward) metric to evaluate the overall context-wise profits of the whole list simulated by the evaluator (*i.e.,*$\text{GRN}_{\mathcal{E}}$), which is calculated as follows:

$$LR(O) = \sum_{x_o^i \in O} \text{E}(x_o^i | u, O). \tag{14}$$

Compared with NDCG, we argue that LR is a more fine-grained metric for the evaluation of rerank, which fully estimates the intra-correlations within the top-k results and the overall profits are evaluated in a more precise manner. Moreover, the generalization capability of the LR metric is much stronger since it can rate the items that users have not browsed yet.

For online A/B testing in **RQ4**, We choose PV and IPV metrics, which are widely adopted in industrial recommender systems for evaluating online performance. Specifically, PV and IPV is defined as the total number of items that users browsed and clicked, 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.

5.1.5 Significance Test. For experimental results in Tables 3, 4, 5 and 6, 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 the generator and evaluator module in GRN (*i.e.,*$\text{GRN}_{\mathcal{G}}$ and $\text{GRN}_{\mathcal{E}}$) and other baselines on two datasets in Table 3 and Table 4. The major findings from the experimental results are summarized as follows:

- On both datasets, point-wise methods (*i.e.,* DNN and DeepFM) achieve relatively pool performance for the task of interaction probability prediction and reranking. It indicates that feature engineering of user and item profile is incapable of covering the list-wise information contained by the input ranking list. Generally, list-wise methods achieves remarkable improvements in most cases. By considering the mutual influence among the input ranking list, these methods learn a refined scoring function aware of the feature distributions of the input ranking list. Among these methods, DLCM and PRM consistently outperform MIDNN in all cases, proving that deep structures (*i.e.,* RNN and self-attention) has superior abilities in extracting the feature distribution compared with handmade feature engineering. Moreover, compared with DLCM, adequate improvements are observed by PRM, which demonstrates the effectiveness of self-attention and personalization in the reranking task.
- $\text{GRN}_{\mathcal{E}}$ consistently yields the best performance on the Loss, AUC and GAUC metrics of both datasets. In particular, $\text{GRN}_{\mathcal{E}}$ improves over the best baseline PRM by 0.073, 0.022 in AUC, and 0.005, 0.006 in GAUC on Rec and Ad dataset, respectively. By taking full advantage of the contextual information in the final ranking list, $\text{GRN}_{\mathcal{E}}$ shows the excellent ability to provide more precise contextual interaction probabilities. Different from extracting information from the disordered input ranking list in DLCM and PRM, with the help of well-designed Bi-LSTM and self-attention mechanism, $\text{GRN}_{\mathcal{E}}$ capture the sequential dependency and mutual influence in the

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

Model	Rec			Ad		
	Loss	AUC	GAUC	Loss	AUC	GAUC
DNN	0.158	0.589	0.931	0.187	0.587	0.848
DeepFM	0.152	0.599	0.933	0.186	0.588	0.848
MIDNN	0.143	0.610	0.936	0.185	0.600	0.848
DLCM	0.138	0.616	0.938	0.185	0.602	0.849
PRM	<u>0.121</u>	<u>0.630</u>	<u>0.942</u>	<u>0.184</u>	<u>0.605</u>	<u>0.850</u>
Seq2Slate†	-	-	-	-	-	-
$\text{GRN}_{\mathcal{E}}$	**0.095***	**0.693***	**0.960***	**0.182***	**0.610***	**0.856***

† Note that Seq2slate is designed to generate the final ranking list directly and can not be used to give predictions on the recorded final ranking list.

Table 4: Overall performance comparison *w.r.t.* the reranking strategy (bold: best; underline: runner-up).

Model	Rec		Ad	
	NDCG@5	LR@5	NDCG@5	LR@5
DNN	0.042	0.164	0.109	0.199
DeepFM	0.043	0.169	0.111	0.203
MIDNN	0.050	0.175	0.117	0.212
DLCM	0.052	0.182	0.119	0.228
PRM	<u>0.054</u>	0.186	<u>0.120</u>	0.232
Seq2Slate	0.050	<u>0.192</u>	0.116	<u>0.235</u>
$\text{GRN}_{\mathcal{G}}$	**0.062***	**0.203***	**0.123***	**0.240***

Table 5: Ablation study of the evaluator (BL: Bi-LSTM; SA: self-attention).

Model	Loss	AUC	GAUC
$GRN_{\mathcal{E}}$(- BL)	0.120	0.632	0.942
$GRN_{\mathcal{E}}$(- SA)	0.102	0.684	0.955
$GRN_{\mathcal{E}}$	**0.095***	**0.693***	**0.960***

Table 6: Ablation study of the generator (EL: evolving layer; AL: Activating layer; DR: different reward; SR: self reward).

Model	NDCG@5	LR@5
Greedy	0.054	0.184
$GRN_{\mathcal{G}}$(- EL)	0.060	0.192
$GRN_{\mathcal{G}}$(- AL)	0.055	0.185
$GRN_{\mathcal{G}}$(- DR)	0.059	0.195
$GRN_{\mathcal{G}}$(- SR)	0.033	0.099
$GRN_{\mathcal{G}}$	**0.062***	**0.203***

final ranking list, which is another essential and effective factor to affect the contextual predictions.

- $GRN_{\mathcal{G}}$ significantly and consistently outperforms state-of-the-arts methods by a relatively large margin on both datasets across all metrics. Overall, on Rec and Ad dataset, $GRN_{\mathcal{G}}$ achieves performance gains over the best baseline (*i.e.,* Seq2Slate) by 0.008 and 0.003 for NDCG , 0.011 and 0.005 for LR, respectively, which evidences that the technically designed model architecture and training procedure of $GRN_{\mathcal{G}}$ have successfully transformed the context-wise ability of $GRN_{\mathcal{E}}$ into production. The evolving and activating layers model the information of the selected list, assisting the selector layer to make the most appropriate choice for each step by comparing in the input ranking list. Besides, under the guidance of $GRN_{\mathcal{E}}$, policy gradient with the proposed advantage reward helps distill the reranking knowledge into $GRN_{\mathcal{G}}$ comprehensively.
- Compared with the experimental results on the Rec dataset, the performance lift on the Ad dataset is relatively slight. One possible reason is that Ad dataset is published with random sampling, resulting in the inconsistent and incomplete list records as well as weaker intra-list relevance of user feedback.

5.3 Study of GRN (RQ2)

In this section, we perform a series of experiments on the Rec dataset to better understand the traits of GRN, including well-designed components of evaluator, generator and training procedure. It is noteworthy that similar trends can also be observed on the Ad dataset, which are omitted due to the page limitation.

5.3.1 Ablation Study of $GRN_{\mathcal{E}}$. By leveraging the contextual information in the final ranking list, $GRN_{\mathcal{E}}$ is designed to predict the context-wise interaction probability more precisely. To examine the effectiveness of in the interacting layer, we prepare two variants of $GRN_{\mathcal{E}}$:

- $GRN_{\mathcal{E}}$(- BL): The variant of evaluator, which removes the Bi-LSTM.
- $GRN_{\mathcal{E}}$(- SA): The variant of evaluator, which removes the self-attention mechanism.

The comparison results of $GRN_{\mathcal{E}}$ with its variants are show in Table 5. The declines of AUC and GAUC metrics are observed after removing each component, which demonstrate their effectiveness for capturing the context-wise information. Besides, $GRN_{\mathcal{E}}$(- BL) performs worse than $GRN_{\mathcal{E}}$(- SA), which means modeling the two-way sequential information of the final ranking list is more important.

5.3.2 Ablation Study of Generator and Training Procedure. Both the model design and training procedure help achieve the best reranking results. In this section, we carefully review the contributions of each component. The variants of generator and training procedure are listed as follows:

- Greedy: Directly reranking in descending order according to the score of evaluator.
- $GRN_{\mathcal{G}}$(- EL): The variant of generator without evolving layer, which ignores the evolution of user intent (*i.e.,* Eq. 5).
- $GRN_{\mathcal{G}}$(- AL): The variant of generator without activating layer, which ignores the influence between items in the input ranking list (*i.e.,* Eq. 6).
- $GRN_{\mathcal{G}}$(- DR): The variant of training procedure, which removes the differential reward (*i.e.,* Eq. 11) in the advantage reward.
- $GRN_{\mathcal{G}}$(- SR): The variant of training procedure, which removes the self reward (*i.e.,* Eq. 10) in the advantage reward.

We summarize the results in Table 6 and have the following observations:

- Reranking based the greedy strategy achieves a relatively poor performance on the on both metrics. It indicates that the $GRN_{\mathcal{E}}$ is capable of capturing context-wise information, while it may not be a good way to ranking directly by scores since the greedy strategy changes each item's context in the final ranking list.
- The performance of $GRN_{\mathcal{G}}$(- EL) and $GRN_{\mathcal{G}}$(- AL) proves the importance of modeling the selected item list for a better choice at each step.
- Intuitively, $GRN_{\mathcal{G}}$(- SR) perform worst due to the lack of item's own reward in the list. $GRN_{\mathcal{G}}$ performs better than $GRN_{\mathcal{G}}$(- DR) by considering the context-wise reward of each item in the final ranking list more comprehensively.

5.4 Case Study (RQ3)

In order to clearly demonstrate how GRN addresses the limitations of the greedy reranking strategy existed in previous methods, we conduct one case study in the large-scale industrial Rec dataset. As shown in Fig. 5, the main findings are summarized as follows:

- The rating scores below the item are estimated by a well-trained point-wise model. Intuitively, according to the score

Figure 5: Illustrative example from Rec dataset, which shows the difference between greedy strategy and our learned reranking strategy.

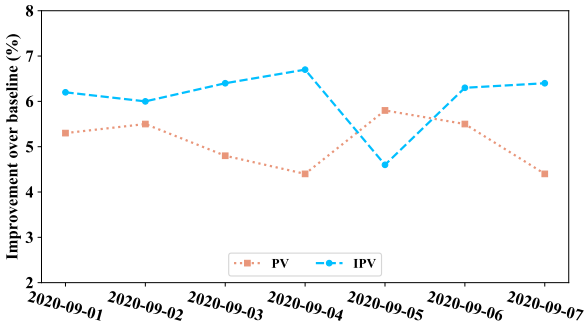

Figure 6: Online performance on the homepage for the *Guess You Like* scenario in Taobao app. y-xis denotes the improvement ratio over the existed deployed baseline.

distribution, we can find that the target user is most interested in clothing category recently, followed by jewelry and pants.

- The greedy reranking strategy choose to place four clothes in its top-5 results. After this user thoroughly browsing and engaging with the first clothe, he/she may be tired of the clothing category and eager for other categories he/she is also interested in. In this case, the greedy reranking strategy can not reach the best recommendation results and further decrease the user experience.
- Different form the greedy reranking strategy, the generator selects the item by considering the contextual items. The recommendation results are more pluralistic, taking more chance to capture and activate the user's latent interests. Hence, the generator shows its ability to generate a more contextually effective and attractive recommendation list, making users engage with the RS more.

5.5 Online A/B Testing (RQ4)

To verify the effectiveness of our proposed framework GRN in the real-world settings, GRN has been fully deployed in homepage for the *Guess You Like scenario*, the most popular recommendation scenario of Taobao application [6]. 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 issue of high latency to the user experience, it brings great challenge to deploy GRN into production.

To this end, we make the following improvements for efficient deployment:

- Instead of the traditional cloud-to-edge framework, we directly deploy GRN on edge as introduced in EdgeRec [15]. In this way, the network latency between the cloud server and user edge is saved. Nearly 30 milliseconds are saved, consisting of 10 requests and 3 milliseconds per request, which could be a bottleneck of deploying a context-wise algorithm.
- Considering the long total time cost, we developed the advance trigger and delay completion mechanism. Specifically, GRN is requested a few places in advance of the last position of current page. Besides, GRN will return more results than expected at each request, to make up for the vacancy of the next request.

After successful deployment of the proposed GRN, we evaluate the performance from "2020-09-01" to "2020-09-07" and report the performance comparison with deployed baseline model PRM in the Fig. 6. Not surprisingly, we observe that GRN consistently and significantly outperforms PRM model across both PV and IPV metrics, which further verifies the effectiveness of our proposed framework GRN. Overall, GRN achieves performance improvement over the best baseline PRM of **5.2**% for PV and **6.1**% for IPV, with the average cost of **32** milliseconds for online inference.

6 CONCLUSION

In this paper, we highlight the importance of modeling the contextual information in the final ranking list and address how to leverage and transform such information for better recommendation. We propose a novel two-stage context-wise framework GRN to learn to rank with contexts. Furthermore, we elaborate on the model design of evaluator and generator in GRN, which aim to predict the contextual interaction probability more precisely and learn a better context-wise reranking strategy, respectively. 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, GRN 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 investigate into how to incorporate the long-term reward for learning a better reranking strategy.

[6] In practice, we only deploy the GRN_G into production for serving reranking task.

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In *ACM Special Interest Group on Information Retrieval*. 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 *ACM Special Interest Group on Information Retrieval*. 85–92.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

[4] 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).

[5] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *International Conference on Machine Learning*. 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 *Conference and Workshop on Neural Information Processing Systems*. 5622–5633.

[8] 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 *ACM Conference on Recommender Systems Workshop*. 7–10.

[9] Covington, Paul, Adams, Jay, Sargin, and Emre. 2016. Deep neural networks for youtube recommendations. In *ACM Conference on Recommender Systems*. 191–198.

[10] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *The International World Wide Web Conference*. 271–280.

[11] 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 *International Joint Conference on Artificial Intelligence*. 2301–2307.

[12] 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).

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

[14] 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.

[15] Yu Gong, Z. Jiang, Kaiqi Zhao, Q. Liu, and Wenwu Ou. 2020. EdgeRec: Recommender System on Edge in Mobile Taobao. In *ACM International Conference on Information and Knowledge Management*.

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

[17] Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[18] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 217–226.

[19] Lei Li, Dingding Wang, Tao Li, Daniel Knox, and Balaji Padmanabhan. 2011. SCENE: A Scalable Two-Stage Personalized News Recommendation System *(ACM Special Interest Group on Information Retrieval)*. Association for Computing Machinery, New York, NY, USA, 125–134. https://doi.org/10.1145/2009916.2009937

[20] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *International Conference on Intelligent User Interfaces*. 31–40.

[21] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. 1928–1937.

[22] 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 *ACM Conference on Recommender Systems*. 3–11.

[23] 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 *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2671–2679.

[24] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. In *The International World Wide Web Conference*. 285–295.

[25] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Conference and Workshop on Neural Information Processing Systems*. 5998–6008.

[28] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *Conference and Workshop on Neural Information Processing Systems*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). 2692–2700.

[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 *ACM International Conference on Information and Knowledge Management*. 2165–2173.

[30] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *ACM Special Interest Group on Information Retrieval*. 287–294.

[31] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *the Association for the Advance of Artificial Intelligence*. 5941–5948.

[32] 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 *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1059–1068.

[33] Tao Zhuang, Wenwu Ou, and Zhirong Wang. 2018. Globally Optimized Mutual Influence Aware Ranking in E-Commerce Search. In *International Joint Conference on Artificial Intelligence*. 3725–3731.