AliExpress Learning-To-Rank: Maximizing Online Model Performance without Going Online

Guangda Huzhang, Zhen-Jia Pang, Yongqing Gao, Yawen Liu, Weijie Shen, Wen-Ji Zhou, Qing Da, An-Xiang Zeng, Han Yu, Yang Yu *Member, IEEE*, Zhi-Hua Zhou *Fellow, IEEE*

Abstract—Learning-to-rank (LTR) has become a key technology in E-commerce applications. Most existing LTR approaches follow a supervised learning paradigm from offline labeled data collected from the online system. However, it has been noticed that previous LTR models can have a good validation performance over offline validation data but have a poor online performance, and vice versa, which implies a possible large inconsistency between the offline and online evaluation. We investigate and confirm in this paper that such inconsistency exists and can have a significant impact on AliExpress Search. Reasons for the inconsistency include the ignorance of item context during the learning, and the offline data set is insufficient for learning the context. Therefore, this paper proposes an evaluator-generator framework for LTR with item context. The framework consists of an evaluator that generalizes to evaluate recommendations involving the context, and a generator that maximizes the evaluator score by reinforcement learning, and a discriminator that ensures the generalization of the evaluator. Extensive experiments in simulation environments and AliExpress Search online system show that, firstly, the classic data-based metrics on the offline dataset can show significant inconsistency with online performance, and can even be misleading. Secondly, the proposed evaluator score is significantly more consistent with the online performance than common ranking metrics. Finally, as the consequence, our method achieves a significant improvement ($>2\%$) in terms of Conversion Rate (CR) over the industrial-level fine-tuned model in online A/B tests.

Index Terms—Learning-To-Rank, evaluation, offline-online inconsistency

1 Introduction

Learning-to-rank (LTR) has been the focus of online search engine and recommender system research for enhancing profitability. In a given scenario, existing LTR approaches generally assume that the click-through rate (or conversion rate) of an item is an intrinsic property of the item, which needs to be accurately discovered with sophisticated predictive models. Under such an assumption, it is reasonable to focus on data-based ranking metrics such as Area Under Curve (AUC) and Normalized Discounted Cumulative Gain (NDCG) to evaluate model performance, which has led to many LTR models closely match the labels in the offline data.

However, practitioners have reported the "offline-online inconsistency problem" in recent works [4], [26], [28], [31]: *a new model achieving significant improvement on offline metrics may not achieve the same improvement when deployed online*. One major reason of the inconsistency is that purchase intention of customers is influenced by the context. Here is an example: if an item is shown together with similar but more expensive items, the likelihood of purchase increases, which is known as the *decoy effect*. Figure 1 illustrates this with an example that context may drastically change behaviors of customers. Another commonly observed phenomenon is that, once a customer clicks an item, the chance of clicking the next item usually decreases [39]. Thus, the inconsistency is introduced since the labels logged from feedback of customers may no longer be true when the order of items changes, i.e. if we present a new order of the same items to the same customer, the same customer may make a different purchase decision. Therefore, we need to carefully consider the context in E-commerce rankings.

To address the issue of the contextual influence, the *re-ranking* strategy has been proposed [45] for practical LTR applications. The ranking system first selects a small set of candidate items; then, it determines the order of these candidate items using a re-ranking model. Different from classic LTR models, re-ranking models can enhance the understanding of the candidate items in a holistic manner. Most of re-ranking models are trained by supervised learning [5], [27], [45], and are evaluated with data-based metrics on offline datasets. However, in context-sensitive scenarios, classic re-ranking models, which follow the supervised learning paradigm, still do not explore the combinatorial space to find the best permutation of items. Thus, these models are less possible to optimize actual performance metrics, such as Conversion Rate (CR) and Gross Merchandise Volume (GMV), after deployment. For example, such models cannot generate creative order (as shown on the right of Figure 1) which can potentially have a better performance.

In order to address the aforementioned problem, we require an evaluation approach beyond the static dataset as well as an exploration approach beyond the supervised learning paradigm. The ideal approach is to learn through interacting with the real customers in the online environ-

- *Guangda Huzhang, Wen-Ji Zhou, Qing Da, and An-Xiang Zeng are with Alibaba.*
- *Zhen-Jia Pang, Yongqing Gao, Yawen Liu, Weijie Shen, Yang Yu, and Zhi-Hua Zhou are with State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China.*
- *Han Yu is with Nanyang Technological University, Singapore.*
- *The first two authors contributed equally to this work. Corresponding authors: Wenji Zhou (Alibaba, eric.zwj@alibaba-inc.com) and Yang Yu (Nanjing University, yuy@nju.edu.cn).*

Manuscript revised XXX XXXX, 2020.

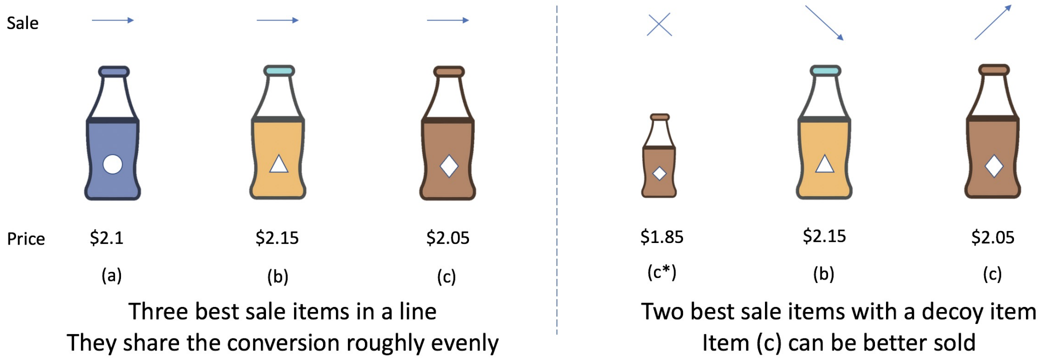
Sale
Price
$2.1
$2.15
$2.05
(a)
(b)
(c)
Three best sale items in a line
They share the conversion roughly evenly
$1.85
$2.15
$2.05
(c*)
(b)
(c)
Two best sale items with a decoy item
Item (c) can be better sold

Fig. 1. An example of *the decoy effect*, or *the anchoring effect*. Standard LTR solutions produce the order on the left as the items are well sold, but the creative orders on the right may achieve better overall performance. Item (c*) plays as a decoy (a smaller but expensive version of item (c)) that will not be purchased, while improves the impression of item (c).

ment. However, such a direct interaction approach can be highly risky and costly as repeated (possibly unsuccessful) trial-and-error may negatively impact customer experience and revenue.

In this paper, we present the Evaluator-Generator Reranking *(EG-Rerank)* framework for E-commerce LTR systems to address the offline-online inconsistency problem while avoiding the pitfalls of direct online interaction-based evaluation. EG-Rerank consists of *an evaluator* and *a generator*. The evaluator aims at modeling user feedback (i.e. predicting the probabilities of purchase given a list). It works as a virtual environment of the real online environment, and its scores can better reflect intentions of customers than labels used in static datasets. The generator is then learned according to the evaluator through reinforcement learning.

In EG-Rerank, we have noticed that the evaluator is learned from the offline labeled data and may not generalize well for generated ranks that are very different from the data. Therefore, we introduce *a discriminator* model to provide a self-confidence score for the evaluation, which is trained adversarially to the generator and guides the generator to produce lists not far from the data. Therefore, the discriminator restricts the exploration space of the generator and ensures a more trustable update. We name EG-Rerank with a discriminator as *EG-Rerank+*.

This work makes the following contributions to E-commerce LTR:

- Through experiments in a simulation environment and AliExpress Search online system, which is one of the world's largest international retail platforms and has more than 150 million buyers from more than 220 countries[1], we demonstrate the significance of the offline-online inconsistency problem.
- We further show that the EG-Rerank evaluator can be a more robust objective compared to existing metrics on offline data, and can serve as a substitution of these metrics.
- We present the EG-Rerank and EG-Rerank+ as two approaches of the evaluator-generator framework for ranking applications.
- In AliExpress Search, EG-Rerank+ consistently improves the conversion rate by 2% over the fine-tuned industrial-level re-ranking model in online A/B tests, which translates into a significant improvement of revenue increasing.

2 Related Work

Learning-to-rank (LTR) solves item ranking problems through machine learning. There are several types of LTR models, including point-wise, pair-wise, list-wise, and so on. Point-wise models [11], [24] treat the ranking problems as classification or regression tasks for each item. Pair-wise models [7], [8], [21] convert the original problem into the internal ranking of pairs. List-wise models [2], [9], [41] use well-designed loss functions to directly optimize the ranking criteria. Group-wise models [3] and page-wise models [43] are proposed recently, which are similar to re-ranking models [45]. However, all the above-cited methods focus on optimizing offline data-based ranking metrics like AUC and NDCG, which can produce offline performance inconsistent with actual online performance.

Slate optimization is a close topic with LTR. Similar to the objective of re-ranking, it also aims to optimize the profit of the whole slate (i.e. a list or a webpage). Recently, there are a few slate optimization studies pay more attention to metrics that consider the new orders. An idea is to combine a generation framework with evaluations on lists to solve slate optimizations. However, multiple sampling [39] and heuristic search (such as beam search [45]) involved in such approaches can be time-consuming for online application. An independent work [42] studies a similar Evaluator-Generator reinforcement learning framework in pure offline environments for CTR predictions, which is a secondary goal in E-commerce. A solution for exact-K recommendation [15] is to imitate outputs with positive feedback through behavior cloning. Compared with such approaches, EG-Rerank+ encourages models to over-perform the experts (offline data) through generative adversarial imitation learning (GAIL) [16], which has been examined to be a better choice for imitation learning [14], [17], [34]. A team from Huawei [40] proposes an alternative offline metric for GMV optimization models.

[1] The information about our platform can be found in https://sell.aliexpress.com/__pc/4DYTFsSkV0.htm

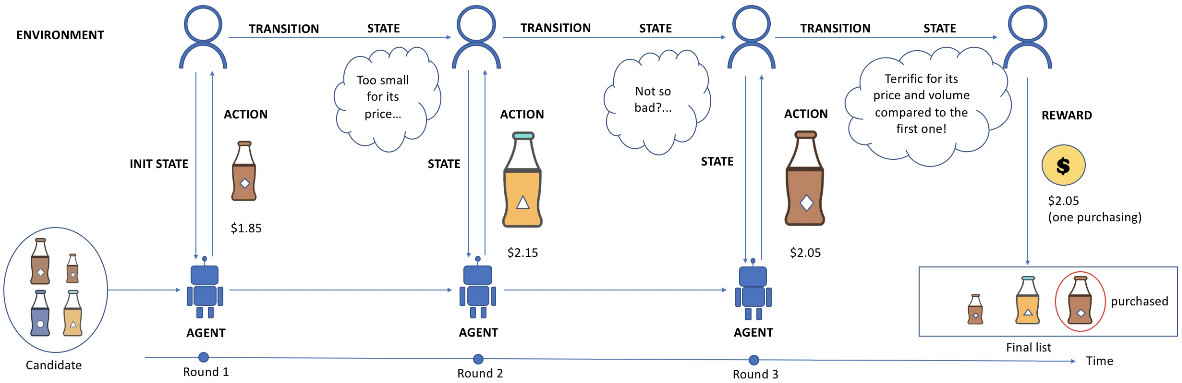

Fig. 2. Illustration of the Markov decision process modeling of E-commerce interactions. The environment state (schematically) describes the intention of a customer at that time, which is influenced by previous interactions.

Reinforcement learning [35] algorithms aim to find the optimal policy that maximizes the expected return. Proximal policy optimization (PPO) [33] is one of them, which optimizes a surrogate objective function by using the stochastic gradient ascent. PPO retains some benefits of region policy optimization (TRPO) [32], but it is much simpler to implement with better sample complexity. Some works [10], [19] use a pure reinforcement learning method for slate optimization and achieved good performance in online environments. Their methods focus on a series of slates and optimize long term value (LTV) of user feedback. Different from their works, we focus on the fundamental challenge to optimize a single slate with only one round of interaction.

The offline-online inconsistency problem has been reported by other studies [4], [26], [28], [31]. Unbiased learning is a related topic to mitigate selection bias. These solutions (e.g., Propensity SVM-Rank [22], SVM PropDCG and Deep PropDCG [1]) can increase the accuracy of trained models. Different from our focus, these models do not consider context and do not attempt to find the best order.

3 Preliminary

3.1 Problem Definition

The objective of a re-ranking model is to find the best permutation of N candidate items. In this paper, we assume that customers browse the list of items displayed in a webpage from top to bottom. Let $o_{1:i}$ denote the arrangement of top i items. When a customer u looks at a list o, the probability that customer u purchases the i-th item can be expressed as $p(C_i|o_{1:i}, u)$, where *the random variable* C_i denotes the event that customer u purchases the i-th item given the layout $o_{1:i}$. Classic LTR models treat $p(C_i|o_{1:i}, u)$ as $p(C_i|u)$, where the items have no mutual influence. Different from these models, re-ranking models have complete knowledge of candidate items.

We use C to denote the number of purchases for the whole list (i.e. the summation of C_i). Then, we have a formula that computes the expected number of purchases for list o and customer u as follows:

$$\mathbb{E}(C|o,u) = \sum_{i=1}^{N} \mathbb{E}(C_i|o,u) = \sum_{i=1}^{N} p(C_i|o_{1:i},u) \tag{1}$$

Given a fixed set of candidate items I and a fixed customer u, the goal is to find a permutation o^* that maximizes the expected number of purchases:

$$o^* = \underset{o \in perm(I)}{\operatorname{argmax}}\ \mathbb{E}(C|o,u) = \underset{o \in perm(I)}{\operatorname{argmax}} \sum_{i=1}^{N} p(C_i|o_{1:i},u) \tag{2}$$

where $perm(I)$ is the set of all permutations of set I.

3.2 Reinforcement Learning Re-ranking

For an ordered list of items, we assume that the customer browses each item from top to bottom and decides whether to purchase or not. Given a candidate set I containing N items and the customer u, we model a re-ranking task as a Markov Decision Process (MDP) with a tuple of $\langle S, A, P, R \rangle$ with discount rate $\gamma = 1$ (also see Figure 2):

- State space S: a state $s_t \in S$ consists of the user feature and an ordered list of selected items before time t. Concretely, we have $s_t = (u, o_t)$ and the observation $o_t = (x_0, x_1, ..., x_k, ..., x_{t-1})$, where x_k is the item selected from the candidate set I at time k. Specially, the initial list $o_0 = ()$ is an empty list.
- Action space A: an action $a_t \in A$ consists of a single item x_t which is selected from the items set $I \setminus \{x_0, x_1, ..., x_{t-1}\}$.
- Reward R: a reward r_t can be written as $r_t = R(s_t, a_t) = R(s_t, x_t) = p(x_t|s_t)$, where $p(x_t|s_t)$ indicates the probability that the customer purchase the item x_t.
- Transition probabilities P: After the action a_t has been chosen (i.e. x_t), the state deterministically transitions to the next state s_{t+1}, where $s_{t+1} = (u, o_{t+1}) = (u, (x_0, x_1, ..., x_{t-1}, x_t))$ so that $P_{s_t, s_{t+1}} = 1$ (and 0 for others).

In our work, we design an evaluator to estimate the purchase probability $p(x_t|s_t)$. The evaluator plays an important role in training, and can be learned by any discriminative machine learning method. The reward, which is given by the evaluator, helps train the generator through reinforcement learning. Models trained by supervised learning can select the short-term best action in each step, such as finding the item that maximizes purchase probability $p(x_t|s_t)$, but the greedy strategy cannot optimize the long-term reward.

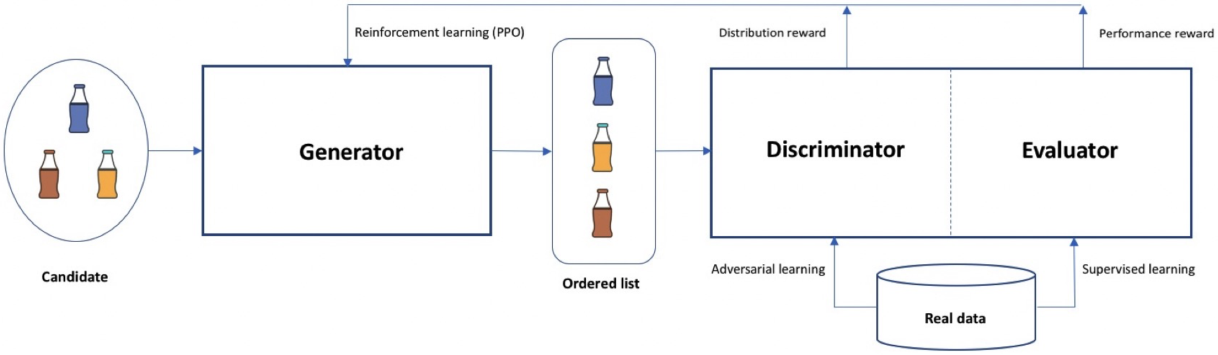

Fig. 3. The architecture of EG-Rerank+. For EG-Rerank, there is no discriminator module. The evaluator is trained and fixed first, and then we can train the generator with rewards provided by the evaluator. The generator and the discriminator are trained simultaneously.

3.3 LTR in Online Systems

A widely accepted view is that people pay more attention to items which appear early in the list. Therefore, it makes sense to design a model to find the best-matched items and put them at the top of a list. Following this guideline, we can reduce the ranking task to a conversion rate prediction task as most LTR models do. However, the conversion rate depends on the context in practical applications. A study [39] has shown when a customer clicks an item (news in their case), the chance of clicking the next item decreases. In situations in which people never click adjacent items, greedy ranking approaches produce improper orders. The above example shows that even an accurate model with high AUC and NDCG scores can perform poorly in some scenarios due to contextual factors. Therefore, *comparing models by data-based metrics in a static offline dataset may be misleading*. It is desirable for E-Commerce LTR research to develop a more robust evaluation for examining models.

4 The Proposed Approach

We propose to use an evaluator-generator framework that trains the generator of lists not from fixed labeled data or optimizing data-based metrics, but learn to generate lists that maximize the evaluator scores.

In this framework, the evaluator needs to evaluate the performance of any given list, and it is expected to closely track the actual online performance. To train a list generator that maximizes the evaluator score, a natural tool is to employ reinforcement learning to transfer gradient information. As a result, we propose the evaluator-generator reranking approach (EG-Rerank). Moreover, to ensure that the evaluator gives trustable scores, we further introduce a discriminator to tell whether the generated list is far from the data. This results in the implemented version, EG-Rerank+, contains an evaluator, a generator, and a discriminator:

- The trained evaluator predicts the performance of the given lists. We use a supervised learning approach.
- The trained generator produces orders with high scores (from the evaluator). We propose a reinforcement learning approach to train it with rewards provided by the evaluator and the discriminator.
- The trained discriminator measures how much the predictions of the evaluator can be trusted. We design an adversarial learning approach with the outputs of the generator and labeled data. Without a discriminator, the evaluator might wrongly evaluate the performance of lists which are much different from the training data.

Figure 3 shows the architecture of EG-Rerank+. Details is described in the following subsections.

4.1 The Evaluator

The evaluator is the key model in EG-rerank. It produces the score, or the reward, for training the generator. The structure of our evaluator is shown in Figure 4. The input includes the features of a list of items $(x_1, x_2, ..., x_N)$ and the scenario feature bg. The scenario feature is independent from the items, and provides additional information such as date, language, and user profiles. Let DNN^k denote a network with k fully connected layers. We use DNN^2 to extract the hidden feature s_i for each item and an LSTM cell to process the contextual state h_i of the first i items:

$$s_i = DNN^2([x_i, bg]), \quad h_i = LSTM(h_{i-1}, s_i) \tag{3}$$

We include another DNN^3 to estimate the conversion rate of item i under state h_i and h_0 is initialized with encoding of candidate set I. In addition, we co-train [25] a click-through rate prediction task to overcome the issue of the sparsity of purchased samples. It helps the model learn common knowledge for predicting clicks and purchases.

$$p_i = \text{sigmoid}(DNN^3([s_i, h_i])) \tag{4}$$

$$p_i^{click} = \text{sigmoid}(DNN^3_{click}([s_i, h_i])) \tag{5}$$

The loss function is a weighted sum of both objectives. The parameter α should be chosen according to the ratio between the number of purchases and the number of clicks. Let label y_i and y_i^{click} be feedback $\{0, 1\}$, and θ be the parameters of the model, we have

$$\begin{aligned} L(\theta) = & \sum \text{cross\_entropy}(p_i, y_i) \\ & + \alpha * \sum \text{cross\_entropy}(p_i^{click}, y_i^{click}) \end{aligned} \tag{6}$$

Requirements for the evaluator: Different from classic conversion prediction models, we want the evaluator to evaluate the quality of lists that closely reflect online performance. We provide several minimal requirements in our

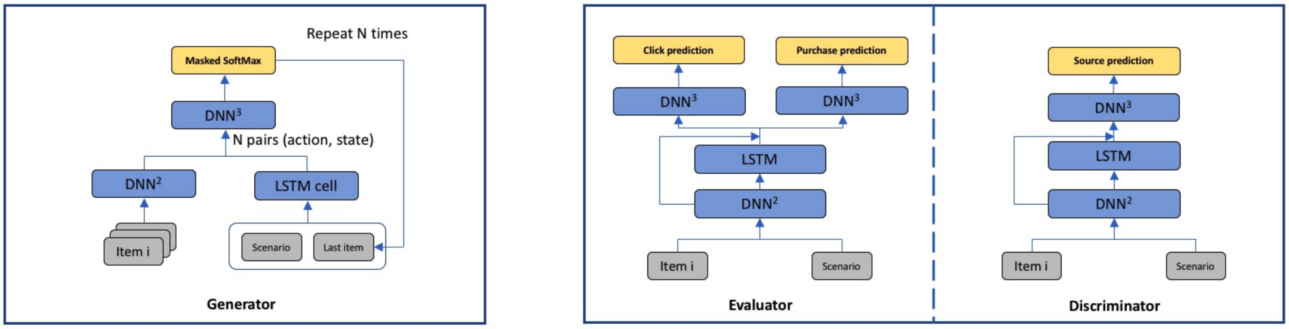

Fig. 4. The network structure of the generator, the evaluator and the discriminator.

experiments. Based on logs of the industrial ranking system, the evaluator should satisfy the following requirements:

- It should be able to identify the better one between a logged list and the reversed list.
- It should be able to identify the better one between a logged list and a random list with the same items.
- It should be able to identify the better one with high confidence between two lists with different labels (e.g., one has a purchase and the other does not).

In our online experiment, the proposed evaluator achieves the first two design requirements with closely to accuracy of 100%. For the third one, it achieves an accuracy of 0.8734, implying that it can identify the better list with high confidence. More details will be discussed in the experimental evaluation section.

4.2 The Generator

With a reliable evaluator, the generator can autonomously explore for the best order. To encode inputs to the generator, we partition the computation into two parts: the feature of the current state and the feature of available actions:

$$\begin{aligned} h_s^{(t)} &= LSTM(h_s^{(t-1)}, DNN^2([bg, x_{out_{t-1}}])) \\ h_a^{(i)} &= DNN^2(x_i) \end{aligned} \tag{7}$$

where $x_{out_{t-1}}$ is the item picked by the user in step $t-1$. Note that $h_a^{(i)}$ is independent from t and can be reused. The output of the encoding process contains N vectors, where $enc_i^{(t)}$ is a combination of a candidate item and a state.

$$enc^{(t)} = [enc_i^{(t)}]_{i=1}^n = [h_a^{(i)}; h_s^{(t)}]_{i=1}^n \tag{8}$$

Then, the generator samples the next action according to $softmax(DNN^3(enc_i^{(t)}))$ for the unpicked item i.

Training algorithm: We use a state-of-the-art reinforcement learning PPO [33] to optimize the generator with feedback from the evaluator. As a recent empirical work revealed, estimating the value of state by multiple sampling is beneficial in a combinatorial space [23]. In our work, we use this strategy to estimate the value and will show its advantage in the experiment section. Concretely, we sample k trajectories (i.e. k generated complete lists) $\tau_i^{s_t}$ with the current policy start from a state s_t. By these trajectories, we can calculate the estimate of state value, denoted as $\widetilde{V}(s_t)$. Other notations can be found in Subsection 3.2.

$$\widetilde{V}(s_t) = \frac{1}{k}\sum_{i=1}^{k}\sum_{(s,a)\in\tau_i^{s_t}} R(s,a) \tag{9}$$

Here $\sum_{(s,a)\in\tau_i^{s_t}} R(s,a)$ is the return, which is also the total number of purchase from the t-th item to the n-th item. Moreover, we apply the standard deviation of value estimation to the loss function to make training more stable. The standard deviation of $\widetilde{V}(s)$ can be formulated as

$$\sigma_{\widetilde{V}}(s_t) = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(\sum_{(s,a)\in\tau_i^{s_t}} R(s,a) - \widetilde{V}(s)\right)^2} \tag{10}$$

The loss function of EG-Rerank is written as

$$L(\theta) = -\hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{11}$$

$$\hat{A}_t = \frac{\sum_{i=t}^{N} R(s_i, a_i) - \widetilde{V}(s_t)}{\sigma_{\widetilde{V}}(s_t)} \tag{12}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{13}$$

Here, $clip(x, l, r)$ is equivalent to $min(max(x, l), r)$. Policy $\pi_{\theta_{old}}$ is the one collecting rewards and π_θ is the current policy. Incorporating $r_t(\theta)$ protects the model from unstable policy changes. A generator generates an order by sampling a trajectory and send it to the evaluator to obtain rewards. Then it updates its parameters with the above loss function.

4.3 EG-Rerank+

All supervised LTR models have a commonly issue: the data we use to train and validate the model is from the offline data, which is collected by a specific system and is generally has a significant bias on selecting samples. It implies that the model has not been well trained in the unseen lists, so it may not give a correct prediction on the unseen lists.

As our evaluator follows the supervised learning paradigm, it also has the above risk. To visualize this issue, we set up a toy task: finding the best permutation of 30 items. A simple rule determines the score of permutations, and the score completely depends on the order of items. We train a model to regress this score with a training set, which is collected by a random but biased strategy. At the same time, we collect a biased validation set. After that, we compute the prediction error of the model in the biased validation set and the unbiased test dataset (which is collected by a random strategy).

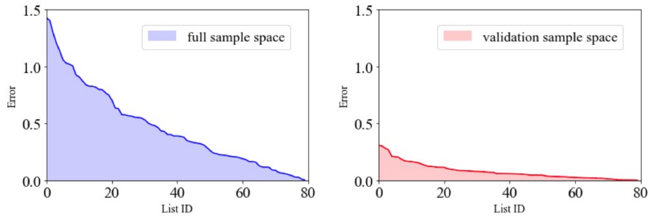

Fig. 5. The error (difference between prediction and ground-truth) in the full sample space (left) and the validation set (right) in a simulation environment. The list with a smaller ID should have a greater error as we sorted them in a decreasing order of error.

The above experiment tries to reproduce the inaccuracy of an evaluator when the input distribution is different from the lists in the training set. For better virtualization, we arrange lists in *a decreasing order* of error as the Figure 5 shows. It is clear to see the error is higher in the unseen distribution.

4.3.1 Design of EG-Rerank+

To ensure that the generator closely mimics the real lists, we introduce a sequential discriminator. The designed discriminator prevents the generator from outputting lists which are too different from the training data (i.e. when the evaluator may provide wrong supervision). With the guidance of the discriminator, the generator will explore solutions in a more reliable and regular space.

Discriminator score $D(x|w)$ represents how a list is possibly from real data as judged by the discriminator. Formally, for an ordered item list $x = (x_1, x_2, ..., x_N)$, the discriminator will score each item, and we let $D(x|w)$ be the summation of scores of items. A sequential structure produces $D(x|w)$ as follows:

$$s_i = DNN^2([x_i, bg]) \tag{14}$$

$$h_i = LSTM(h_{i-1}, s_i) \tag{15}$$

$$score_i = DNN^3([s_i, h_i]) \tag{16}$$

$$D(x|w) = \sum score_i \tag{17}$$

We aim to train the discriminator to distinguish generated lists x from real lists x'. We want to maximize the below expectation and the gradient has the form

$$\hat{\mathbb{E}}_x \left[\nabla_w \log\left(D(x|w)\right)\right] + \hat{\mathbb{E}}_{x'} \left[\nabla_w \log\left(1 - D(x'|w)\right)\right] \tag{18}$$

Finally, we take the output of the discriminator as part of the reward for learning under EG-Rerank+ with an adjustable parameter β:

$$R^+(s_i, a_i) = R(s_i, a_i) + \beta * score_i \tag{19}$$

4.3.2 Advantages of EG-Rerank+

To reveal the advantage of EG-Rerank+, we set up an auxiliary experiment by real data. We collect thousands of real lists from our online system, where all lists are generated from the same search query "phone screen protectors". Figure 6 visualizes the distribution of real lists, outputs of EG-Rerank, and outputs of EG-Rerank+. We plot Figure 6 with t-SNE for data dimensionality reduction, in which the sources (i.e. generated from which model) are not revealed. To reduce the noise from the online environment, we remove 20% records which are farthest away from the centroid in their groups.

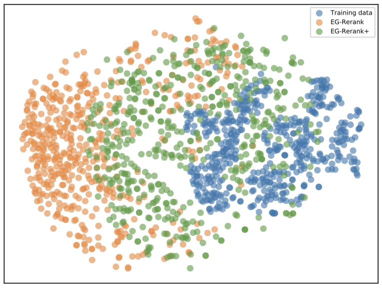

Fig. 6. The distribution of real lists and generated lists.

We can observe that outputs of EG-Rerank+ are closer to the real lists produced by a mature system than the outputs of EG-Rerank. The discriminator idea is a strategy to guide the exploration without having to engage in costly online interactions. In practice, it is possible to deploy online exploration strategies to calibrate the evaluator or the generator. Compare to these methods, EG-Rerank+ is easy to implement, poses no risk to customer experience, and can be examined in offline environments.

5 Experimental Evaluation

For re-ranking tasks, an evaluation based on implicit feedback from existing datasets may lead the inconsistency problem: offline data can hardly reflect the accurate feedback in an unseen distribution. Therefore, a *dynamic judgement* which can produce feedback for every possible output from models needs to be introduced. We consider the two examinations that satisfy the *dynamic* condition:

- *Examination in the simulation environment*. To the best of our knowledge, it is the only offline choice to roughly evaluate model performance. With well-defined simulation protocols, the experimental results can be easily reproduced.
- *Examination in AliExpress Search online system*. Although online trials are expensive and not reproducible, online A/B testing is the only gold standard of evaluation.

In the following contents, we conduct our models in both of them and analyze two research questions. **(RQ1)** Does the evaluation in the static dataset evaluation mislead the learning? **(RQ2)** Does EG-Rerank+ achieve the best performance among existing re-ranking algorithms?

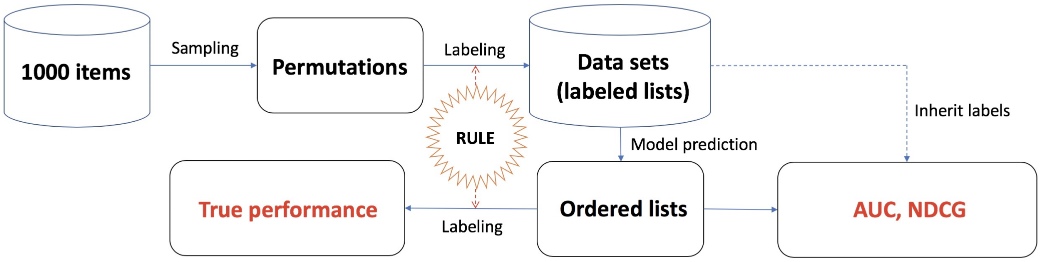

Fig. 7. The workflow of the simulation environment. An extra interaction with simulator RULE is necessary to get the true performance.

5.1 Experiment in a Simulation Environment

We demonstrate that classic data-based metrics are inconsistent with our objective (e.g. the number of conversion) even in a simple LTR scenario with mutual influence information between items. The offline experiments is conducted in a simulation environment[2].

5.1.1 The Simulation Environment

Our simulation environment borrows ideas from the design of RecSim [18]. We prepare $1,000$ items each containing a random feature of length 30, and sample $400,000$ lists of size 15 uniformly from the $1,000$ items. The simulator, which acts as referee in the environment, labels these sampled lists. Then, we divide these lists into two parts of $300,000$ and $100,000$ as the training data and the testing data. To get the true performance, we need an extra interaction with the simulator as Figure 7 shows.

Let L be a list and L_i be the i-th item in list L. Each item receives two scores from the simulator (the "RULE" in Figure 7):

$$f(L_i, L) = \alpha_i r(L_i) + \beta_i g(L_{1:i}) \tag{20}$$

The base conversion rate $r(L_i) \in [0,1]$ of item L_i is computed using a randomly weighted DNN. For mutual influence $g(L_{1:i})$, we consider the similarity between features of selected items. Intuitively, it is better to avoid adjacently displaying similar items. In our experiment, mutual influence g deploys 1 minus cosine distance in the range of $[0,1]$, between L_i and the average of $L_{1:i}$. In addition, we let constant α_i decrease and constant β_i increase according to position i and $\alpha_i + \beta_i = 1$. Therefore, score $f(L_i, L)$ must be a real number in $[0,1]$ and each item receives a $\{0,1\}$ label by sampling with the probability $f(L_i, L)$. We regard the label 1 as a purchase event on an item.

The score of a list, which is the summation of scores of all items, equals the expected number of purchases. We denote this score as the *true score* on a list, and regard the true score as the main objective of a model. Besides, we use labels (collected by old orders) to compare the accuracy of data-based metrics.

In a real environment, top-ranked items receive more attention from customers. We calculate the average conversion rate for each position in our simulation environment with training data. The result is that the conversion rate decreases from 0.5 to 0.3 as the position increases. The property motivate us to display the best items in the top as we do in online systems. However, most of previous simulation experiments (such as [20], [18], and [28]) did not promise the above important condition. It brings unfairness for ranking methods which place the best items in the top.

5.1.2 Comparison Baselines

We arrange the models in several groups:

- **Non-deep methods**. We use SVM-Rank [22] and LambdaMART [8] to represent SVM and boosting LTR methods. We further add Propensity SVM-Rank [22] to examine benefits of de-biasing,.
- **Point-wise methods**. We use miDNN [45] and apply mean square root error (MSE) loss, cross entropy (CE) loss and hinge loss on it.
- **Pair-wise methods**. We apply pair-wise logistic loss and hinge loss on miDNN to represent pair-wise methods which follow RankNet [7]. RankNet* is the pair-wise method we deployed online.
- **List-wise methods**. We include ListNet [9], ListMLE [2], and SoftRank [36] to demonstrate the performance of list-wise approachings.
- **Group-wise methods**. We examine $GSF(5)$ and $GSF(10)$ with the sampling trick introduced in the group-wise scoring framework paper [3] .
- **Advanced Re-ranking**. We add a pointer network solution seq2slate [5], [37] with cross entropy loss into our experiment, and a industial re-ranking method PRM with a random initialization [27].
- **Evaluator-Generator**. GreedyE is a greedy strategy to pick the item which can produce the maximal evaluator score immediately. DirectE uses the evaluator as a classic LTR model to generate lists. Deep Q Network (DQN) is a traditional Q-learning algorithm, and Monte Carlo Control is a substitute with samplings which brings less bias and more computations. DQN can be regarded as a degeneration version of SlateQ [19] with $k = 1$ without involving a CTR prediction (in our testing, it is harmful to model performance). CTR-AC is a recent work that follows REINFORCE [42]. To enhance it, we apply the advantage function to CTR-AC as CTR-AC+. EG-Rerank and EG-Rerank+ (our solutions), the advanced function are computed by sampling instead of using V model in PPO.
- **Evaluators**. Note *all involved evaluators are trained by offline data*, and play as an environment in the training phase of the above methods.

[2] The code of the environment are released in github https://github.com/gaoyq95/RerankSim

Method groups	Method	GAUC	NDCG	Evaluator score	True score
Non-deep methods	SVM-Rank	0.93948±0.00002	0.96826±0.00002	5.56269±0.00004	5.57967±0.00011
	Prop SVM-Rank	0.93661±0.00001	0.97079±0.00001	5.56684±0.00001	5.58434±0.00001
	LambdaMART	0.93188±0.00154	0.97697±0.00106	5.58462±0.00765	5.59179±0.00827
Point-wise methods	miDNN (MSE loss)	0.94767±0.00006	0.97266±0.00007	5.59288±0.00230	5.53829±0.00304
	miDNN (CE loss)	0.94764±0.00006	0.97282±0.00004	5.61278±0.00462	5.56168±0.00572
	miDNN (Hinge loss)	0.93957±0.00114	0.96863±0.00063	5.56862±0.00826	5.53250±0.00616
Pair-wise methods	RankNet (Logistic loss)	0.94724±0.00002	0.97249±0.00002	5.64835±0.00153	5.59949±0.00110
	RankNet* (Hinge loss)	0.94813±0.00003	0.97302±0.00003	5.59475±0.00355	5.54473±0.00291
List-wise methods	ListNet	0.94770±0.00004	0.97279±0.00003	5.62637±0.00374	5.57817±0.00363
	ListMLE	0.94712±0.00004	0.97261±0.00003	5.60474±0.00119	5.56719±0.00155
	SoftRank	0.94728±0.00007	0.97270±0.00004	5.60824±0.00355	5.56788±0.00101
Group-wise methods	GSF(5)	0.94792±0.00003	0.97286±0.00001	5.61773±0.00221	5.56135±0.00255
	GSF(10)	0.94807±0.00003	0.97282±0.00004	5.61838±0.00311	5.56526±0.00335
Advanced methods	seq2slate	0.64042±0.00063	0.75267±0.00057	5.63576±0.00326	5.62322±0.00358
	PRM	0.94787±0.00003	0.97283±0.00002	5.78159±0.00050	5.71738±0.00068
Evaluator-Generator	DirectE	0.96365±0.00029	0.98099±0.00014	5.63850±0.00196	5.59867±0.00228
	GreedyE	0.92918±0.00081	0.96388±0.00045	5.77470±0.00353	5.73948±0.00327
	DQN (SlateQ)	0.71726±0.03948	0.79007±0.01662	6.18838±0.07606	6.19378±0.05797
	Monte Carlo Control	0.77541±0.00957	0.81201±0.01019	6.36602±0.06473	6.29952±0.04813
	CTR-AC	0.56332±0.07063	0.72339±0.04117	5.61957±0.05035	5.76285±0.04597
	CTR-AC+	0.48923±0.05965	0.65473±0.02030	7.03191±0.03798	6.71545±0.02517
	PPO	0.74747±0.00691	0.76107±0.00530	7.10018±0.02135	6.78382±0.01566
	EG-Rerank	0.55774±0.00442	0.69925±0.00279	7.36499±0.01294	6.96224±0.00972
	EG-Rerank+	0.51376±0.01181	0.69171±0.00338	**7.36847±0.01341**	**6.97283±0.01240**

TABLE 1
Models performance in the simulation environment.

5.1.3 Result analysis

We take four indicators (NDCG, GAUC, Evaluator Score and True score) into consideration:

- *GAUC (offline) and NDCG* are two commonly used ranking metrics.
- *Evaluator score* is the prediction from the evaluator.
- *True score* is the goal that models aim to maximize. It is produced by the simulator as we described.

The experiments are repeated 10 times with independent data. We display the mean performance and standard deviation in the Table 1.

Metric accuracy: In the above table, GAUC and NDCG are consistent with each other, but they are inconsistent with the evaluator score as well as the actual performance. Concretely, we enumerate each pair of metrics and compute normalized Kendall Tau Distance as Table 2 shows.

	GAUC	NDCG	Evaluator score	True score
GAUC	-	**0.084**	0.645	0.718
NDCG	**0.084**	-	0.658	0.730
Evaluator score	0.645	0.658	-	**0.099**
True score	0.718	0.730	**0.099**	-

TABLE 2
Normalized Kendall Tau Distance results for each pair of metrics.

The result shows that GAUC and NDCG can only predict the better method between two candidates with a poor probability (less than 30%) by the definition of Normalized Kendall Tau Distance. Instead of that, evaluator can correctly decide the better method with a high probability (greater than 90%). This fact is again provide a strong evidence to **RQ1** that data-based metrics may mislead the learning, and supports that evaluator score is potentially a proper metric for E-commerce LTR model examination.

Performance of Supervised Learning Methods: Classic supervised learning methods get high NDCG and GAUC (more than 0.9) in the simulation environment, which implies they can accurately recover the labels in offline data. However, they have poor true score. In contrast, most of methods with the Evaluator-Generator framework achieve lower NDCG and GAUC, but output much better lists according to the true score metric. This fact is also a positive example for **RQ1**.

Performance of Advanced Methods: Unbiased methods like propensity SVM-Rank, as well as sophisticated reranking methods such as seq2slate and PRM, can bring a small improvement on the true score, but it is not competitive with methods follow Evaluator-Generator frameworks. We also include reinforcement learning solutions such as CTR-AC to show our competitiveness. Therefore, together with the above paragraph, we partially answer **RQ2**: in our simulation experiment, EG-Rerank+ outperforms existing re-ranking methods. On the other hand, traditional $Q(s, a)$ and $V(s)$ are difficult to estimate in this ranking task. The comparison between DQN and Monte Carlo Control, as well as the comparison between PPO and EG-Rerank, implies that replacing the value function with sampling can greatly improve the performance. Benefiting from PPO and sampling, our EG-Rerank (and also EG-Rerank+) steadily outperforms slateQ, CTR-AC, and their variants. Not only in ranking tasks, we conjecture the similar phenomenons exist in other combinatorial optimization scenarios.

Fact 1. *Roughly, EG-Rerank+ are expected to output top* 0.2% *lists in our simulation experiment.*

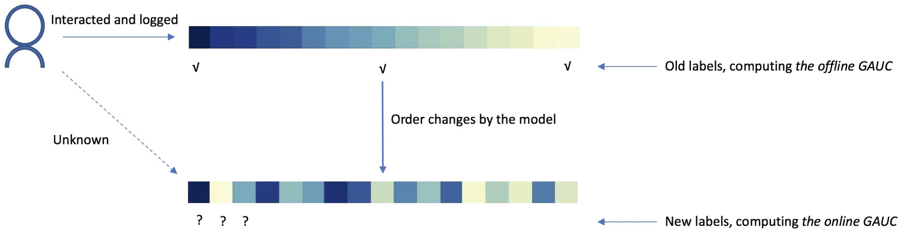
Interacted and logged
√ √ √
Old labels, computing *the offline GAUC*
Order changes by the model
Unknown
? ? ?
New labels, computing *the online GAUC*

Fig. 8. The workflows of computing *offline GAUC* and *online GAUC*, respectively. Colored squares appear as items.

Proof. Consider that we arrange the lists in a interval $[0, 1]$, where adjacent lists have the same spacing. We can use the quantile from 0 to 1 (from worst to best) to represent the ranking. Then the ranking of a sampled list can be approximated by the uniform distribution in $[0, 1]$. From the result table 1, we find that EG-Rerank+ has the similar performance as ENUMERATE-500 does. Therefore, we only need to estimate the expected ranking of the above best list as $\mathbb{E}[max(X_1, X_2, ..., X_n)]$, where X_i is the random variable of the ranking of the i-th sampled list.

Then we can write the expectation in the integral form:

$$\begin{aligned}\mathbb{E}[max(\{X_i\})] &= \int_0^1 \mathbb{E}[max(\{X_i\}) = x]dx \\ &= \int_0^1 x \times p(max(\{X_i\}) = x)dx \\ &= \int_0^1 x \times nx^{n-1}dx = \frac{n}{n+1}\end{aligned} \tag{21}$$

Here probability density function $p(max(\{X_i\}) = x) = \sum_{i=1}^{n} p(X_i = x, X_{j\neq i} < x) = nx^{n-1}$. Replacing n with 500 into Equation (21) yields the desired result. □

Expectation Analysis for Searching in the Combinatorial Space: We set up an additional experiment to evaluate the quality of the output list of EG-Rerank+. Consider an inefficient algorithm ENUMERATE-k: we uniformly sample k lists and use the evaluator to decide and output the best list among them. Without limitation of time, this algorithm can even generate the global optimal list from the view of evaluator (i.e. enumerate all permutations and select the best one). We plot a curve to find the proper k such that EG-Rerank+ has a similar performance with ENUMERATE-k. Then, we can approximate the combinatorial space searching ability of EG-Rerank+ in expectation. In Figure 9, we can see EG-Rerank+ and ENUMERATE-500 have almost the same performance.

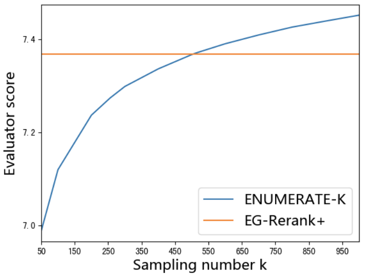

Fig. 9. Performances of ENUMERATE-k with different k. Due to its low efficiency, we plot the curve by a few piece-wise linear functions.

6 Online Cases and Experiments

In the following two subsections, we first share two real cases of AliExpress Search to show the existing inconsistencies between data-based ranking metrics and conversion rate (CR). In our work, CR is the number of purchases divided by the number of arriving customers. Group AUC [44] (GAUC) is commonly used in ranking scenarios when scores of items in different lists are not comparable. GAUC only considers the item pairs in the same list (group), and can be computed as $\frac{1}{|L|}\sum_{l\in L} AUC_l$, where L is the set of testing lists. In our work, we focus on the purchase events so that only purchased items have positive labels. For a fair comparison, pages with no purchase will not be included for GAUC computing (in E-commerce, there are so many lists without a purchase). We consider two types of GAUC as follows (also see Figure 8):

- Offline GAUC: the one computed *before* a model changes item order (i.e. using the old labels).
- Online GAUC: the one computed *after* a model changes item order (i.e. using the new labels).

The GAUC computed in training and validation set evaluation, which uses the labels in old orders, is offline GAUC. It is commonly used in the LTR research and applications. Intuitively, online GAUC is an alternative to the offline GAUC. Online GAUC needs the new feedback from users after model changes the order. These two metrics have a gap that should not be ignored *unless we assume the behaviors of customers on items will keep unchanged after the order changes*. In the following section, we will show both of them are problematic when examining models.

6.1 Inconsistency between Offline GAUC and CR

Table 3 shows the offline GAUC of two ranking strategies during a week. RankNet* [7] is an industrial-level fine-tuned pair-wise model, and it follows the design of RankNet and has the best online performance in long-term experiments.

From the Table 3, we observe that EG-Rerank achieves poor offline GAUC, but greatly improves the conversion rate by more than 2%. This is again a strong evidence to answer **RQ1** and blindly encouraging models to put the best items on the top (as the first strategy does) also cannot achieve consistently good performance.

6.2 Inconsistency between Online GAUC and CR

Table 4 shows the online GAUC and actual performance of three deployed strategies in the past. Without knowing them, strategy 3 appears to be the best one according to

Re-ranking	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Offline GAUC
RankNet*	6.11%	7.17%	8.22%	**8.15%**	4.19%	6.78%	7.84%	5.95%	4.17%	5.16%	4.18%	4.70%	3.52%	7.34%	**0.783**
EG-Rerank	**6.57%**	**9.54%**	**10.45%**	7.98%	**5.70%**	**10.21%**	**10.08%**	**9.02%**	**7.37%**	**6.53%**	**6.00%**	**7.56%**	**6.84%**	**10.65%**	0.512

TABLE 3
Offline GAUC and online performance of models. Each column "Day i" describe the conversion rate gap after we add a re-ranking method.

online GAUC. However, it is problematic and actually the worst strategy due to its low conversion rate. The result clearly shows that online GAUC also cannot reflect actual model performance when deployed online.

Online GAUC	Day 1	Day 2	Day 3	Day 4
Strategy 1	0.786	0.787	0.794	0.792
Strategy 2	0.788	0.786	0.789	0.793
Strategy 3	0.805	0.803	0.800	0.797
Conversion Rate Gap	Day 1	Day 2	Day 3	Day 4
Strategy 1	0.00%	0.00%	0.00%	0.00%
Strategy 2	0.72%	-0.36%	-0.12%	-0.49%
Strategy 3	-8.45%	-8.25%	-7.25%	-10.13%

TABLE 4
Online GAUC and performances during four days.

The above counter-intuitive phenomenon happens since the i.i.d condition has been violated: different models have different distributions of output lists, which are not compatible at all. Consider such an example: a model always picks the best item from its view, and then adds lots of irrelevant items to the list. In this case, the first item receives a high score and the irrelevant items receive low scores from the model. When no one purchases irrelevant items in such a list, the model achieves high online AUC, MAP, and NDCG, but performs poorly in terms of conversion rate.

This experiment reinforces the importance of the proposed evaluator: it is not only a module in our framework, but also *a novel metric for evaluating rankings* which more closely reflects online performance than existing metrics and can solve the issue proposed in **RQ1**.

6.3 Online A/B tests

We set up a few online A/B tests on AliExpress Search. All re-ranking methods are deployed when customers use keyword searching, which is also one of the main composition of the flow in our platform. In our long-term trials, fined-tuned RankNet* has been proven to achieve the best online performance among existing re-ranking methods. Models can access offline data in the last two weeks for training, and the offline data contains $O(10^8)$ displayed lists and $O(10^6)$ purchase records. In addition, we hold a latency testing for different strategies under a high load condition. The latency results are shown in table 5.

	No Re-rank	RankNet*	EG-Rerank
Average latency	106ms	113ms	115ms

TABLE 5
Testing latency. They are held in the same server.

In each A/B test, two models serve a non-overlapping random portion of search queries. Each model needs to serve $O(10^6)$ users and output $O(10^7)$ pages per day. Since the online environment varies rapidly, to get reliable results, all A/B tests are held for more than a week to reduce the variance. We compute the mean and standard deviation during A/B testing. The result is shown in Table 6.

Methods	Online GAUC	Evaluator*	CR gap
No Re-rank	0.758 ± 0.004	+0.00%	+0.00%
RankNet*	0.789 ± 0.002	+15.2%	+5.96% ± 0.017
RankNet*	0.793 ± 0.002	+0.00%	+0.00%
EG-Rerank	0.783 ± 0.003	+9.94%	+2.22% ± 0.011
EG-Rerank	0.774 ± 0.015	+0.00%	+0.00%
EG-Rerank+	0.786 ± 0.003	+0.81%	+0.63% ± 0.008

TABLE 6
Online performance. In Conversion Rate Gap columns, the first row is the baseline.

Here, "Evaluator*" denotes the score of the evaluator model which predicts model performance before the A/B tests start, and it is independent with the one used in EG-Rerank. We can see that the evaluator score is a more consistent and accurate metric than online GAUC. The satisfying performance of EG-Rerank+ also provides an positive answer to **RQ2**.

7 Conclusions

In E-commerce ranking tasks, many studies have lost connections with the real-world, but only focused on offline measurements [38]. To evaluate and improve actual performance, we propose the evaluator-generator framework for E-commerce and EG-Rerank+ for CR optimizations. We demonstrate the evaluator score is a more objective metric which leads to improvements in online situations. Through online testing in a large-scale e-commerce platform, the proposed framework has improved the conversion rate by 2%, which translates into a significant among of revenue.

References

[1] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. A general framework for counterfactual learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 5–14, 2019.

[2] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. Learning a deep listwise context model for ranking refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 135–144. ACM, 2018.

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

[4] Joeran Beel, Marcel Genzmehr, Stefan Langer, Andreas Nürnberger, and Bela Gipp. A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. In *Proceedings of the international workshop on reproducibility and replication in recommender systems evaluation*, pages 7–14, 2013.

[5] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Huaihsin Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. Seq2slate: Re-ranking and slate optimization with rnns. *CoRR*, abs/1810.02019, 2018.

[6] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 193–200, 2006.

[7] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005)*, pages 89–96, 2005.

[8] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[9] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 129–136, 2007.

[10] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne 11-15, 2019*, pages 456–464, 2019.

[11] David Cossock and Tong Zhang. Statistical analysis of bayes optimal subset ranking. *IEEE Trans. Information Theory*, 54(11):5140–5154, 2008.

[12] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019.

[13] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017.

[14] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[15] Yu Gong, Yu Zhu, Lu Duan, Qingwen Liu, Ziyu Guan, Fei Sun, Wenwu Ou, and Kenny Q. Zhu. Exact-k recommendation via maximal clique optimization. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 617–626, 2019.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal*, pages 2672–2680, 2014.

[17] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4565–4573, 2016.

[18] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847*, 2019.

[19] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. Slateq: A tractable decomposition for reinforcement learning with recommendation sets. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, August 10-16, 2019*, pages 2592–2599, 2019.

[20] Ray Jiang, Sven Gowal, Yuqiu Qian, Timothy A. Mann, and Danilo J. Rezende. Beyond greedy ranking: Slate optimization via list-cvae. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[21] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142, 2002.

[22] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 781–789, 2017.

[23] Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

[24] Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 897–904, 2007.

[25] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1137–1140, 2018.

[26] Sean M McNee, John Riedl, and Joseph A Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 extended abstracts on Human factors in computing systems*, pages 1097–1101, 2006.

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

[28] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720*, 2018.

[29] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

[30] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[31] Marco Rossetti, Fabio Stella, and Markus Zanker. Contrasting offline and online results when evaluating recommendation algorithms. In *Proceedings of the 10th ACM conference on recommender systems*, pages 31–34, 2016.

[32] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, pages 1889–1897, 2015.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[34] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4902–4909, 2019.

[35] RS Sutton and AG Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[36] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 77–86, 2008.

[37] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[38] Kiri Wagstaff. Machine learning that matters. *arXiv preprint arXiv:1206.4656*, 2012.

[39] Fan Wang, Xiaomin Fang, Lihang Liu, Yaxue Chen, Jiucheng Tao, Zhiming Peng, Cihang Jin, and Hao Tian. Sequential evaluation and generation framework for combinatorial recommender system. *CoRR*, abs/1902.00245, 2019.

[40] Liang Wu, Diane Hu, Liangjie Hong, and Huan Liu. Turning clicks into purchases: Revenue optimization for product search in e-commerce. In *SIGIR*, pages 365–374. ACM, 2018.

[41] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 1192–1199, 2008.

[42] Junqi Zhang, Jiaxin Mao, Yiqun Liu, Ruizhe Zhang, Min Zhang, Shaoping Ma, Jun Xu, and Qi Tian. Context-aware ranking by constructing a virtual environment for reinforcement learning. In *CIKM*, pages 1603–1612. ACM, 2019.

[43] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 95–103. ACM, 2018.

[44] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1059–1068. ACM, 2018.

[45] Tao Zhuang, Wenwu Ou, and Zhirong Wang. Globally optimized mutual influence aware ranking in e-commerce search. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 3725–3731, 2018.

Guangda Huzhang received the PhD degree from Nanyang Technological University, Singapore, in 2018. After that, he joined Alibaba and worked as a Machine Learning Engineer up to now.

Zhen-Jia Pang received the master degree from Nanjing University, China, in 2020. After that, he joined Huawei and worked as a Machine Learning Engineer up to now.

Yongqing Gao is a master's degree candidate in Nanjing University, China. His current research interests include Reinforcement learning and Recommendation system.

Yawen Liu is a Master student in Nanjing University, he focuses on Deep Reinforcement Learning, as well as its applictions in Autonomous Driving and Recommendation System.

Weijie Shen is a Master student in Nanjing University, his research interests include reinforcement learning, imitation learning as well as their applications in Autonomous Driving and Recommendation System.

Wen-Ji Zhou received the BSc and MSc degree in computer science from Nanjing University, China, in 2016 and 2019, respectively. He joined the Alibaba-INC in June 2019 and worked as a Machine Learning Engineer up to now.

Qing Da received the BSc and MSc degrees in computer science from Nanjing University, China, in 2010 and 2013 respectively. He is currently a senior staff algorithm engineer in the search algorithm team of the Department of International AI at Alibaba Group. His research interests are reinforcement learning and applications of machine learning.

An-Xiang Zeng is a Senior Staff Algorithm Engineer and Director of Alibaba. He is the Head of the International Search and Recommendation of Alibaba. He is pursuing his PhD in Nanyang Technological University, Singapore. He has been working in the search and recommendation field for more than 10 years. His research focuses on search, recommendation and reinforcement learning. He has published more than 10 research papers in leading international conferences and journals.

Han Yu is a Nanyang Assistant Professor in the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He held the prestigious Lee Kuan Yew Post-Doctoral Fellowship from 2015 to 2018. He obtained his PhD from the School of Computer Science and Engineering, NTU. His research focuses on federated learning and algorithmic fairness. He has published over 150 research papers and book chapters in leading international conferences and journals. He is a co-author of the book *Federated Learning* - the first monograph on the topic of federated learning. His research works have won multiple awards from conferences and journals.

Yang Yu received the BSc and PhD degree in computer science from Nanjing University, China, in 2004 and 2011, respectively. He joined the Department of Computer Science & Technology at Nanjing University as an Assistant Researcher in 2011, and is currently Professor of the School of Artificial Intelligence. He has co-authored the book Evolutionary Learning: Advances in Theories and Algorithms, and published more than 40 papers in top-tier international journals and and conference proceedings. He has been recognized as a AI's 10 to Watch by IEEE Intelligent Systems (2018), PAKDD Early Career Award (2018), CCF-IEEE CS Early Career Young Scientist (2020), and was invited to give an IJCAI'18 Early Career Spotlight. He co-founded the Asian Workshop on Reinforcement Learning.

Zhi-Hua Zhou (S'00-M'01-SM'06-F'13) received the BSc, MSc and PhD degrees in computer science from Nanjing University, China, in 1996, 1998 and 2000, respectively, all with the highest honors. He joined the Department of Computer Science & Technology at Nanjing University as an Assistant Professor in 2001, and is currently Professor, Head of the Department of Computer Science and Technology, and Dean of the School of Artificial Intelligence; he is also the Founding Director of the LAMDA group. His research interests are mainly in artificial intelligence, machine learning and data mining. He has authored the books Ensemble Methods: Foundations and Algorithms, Evolutionary Learning: Advances in Theories and Algorithms, Machine Learning (in Chinese), and published more than 150 papers in top-tier international journals or conference proceedings. He has received various awards/honors including the National Natural Science Award of China, the IEEE Computer Society Edward J. McCluskey Technical Achievement Award, the PAKDD Distinguished Contribution Award, the IEEE ICDM Outstanding Service Award, the Microsoft Professorship Award, etc. He also holds 24 patents. He is the Editor-in-Chief of the Frontiers of Computer Science, Associate Editor-in-Chief of the Science China Information Sciences, Action or Associate Editor of the Machine Learning, IEEE Transactions on Pattern Analysis and Machine Intelligence , ACM Transactions on Knowledge Discovery from Data, etc. He served as Associate Editor-in-Chief for Chinese Science Bulletin (2008- 2014), Associate Editor for IEEE Transactions on Knowledge and Data Engineering (2008-2012), IEEE Transactions on Neural Networks and Learning Systems (2014-2017), ACM Transactions on Intelligent Systems and Technology (2009-2017), Neural Networks (2014-2016), etc. He founded ACML (Asian Conference on Machine Learning), served as Advisory Committee member for IJCAI (2015-2016), Steering Committee member for ICDM, PAKDD and PRICAI, and Chair of various conferences such as Program co-chair of AAAI 2019, General co-chair of ICDM 2016, and Area chair of NeurIPS, ICML, AAAI, IJCAI, KDD, etc. He was the Chair of the IEEE CIS Data Mining Technical Committee (2015-2016), the Chair of the CCF-AI (2012-2019), and the Chair of the CAAI Machine Learning Technical Committee (2006-2015). He is a foreign member of the Academy of Europe, and a Fellow of the ACM, AAAI, AAAS, IEEE, IAPR, IET/IEE, CCF, and CAAI.