From Theory to Practice: Testing Harvesting-Based Propensity Estimators on Counterfactual Learning-To-Rank from Implicit Feedback for E-commerce Search

Parantapa Goswami*
Andrés Hoyos-Idrobo*
parantapa.goswami@rakuten.com
andres.hoyosidrobo@rakuten.com
Rakuten Institute of Technology,
Rakuten Group, Inc.

Théo Deschamps†
tdeschamps@kameleoon.com
Kameleoon

Nicolas Feuillet
nicolas.feuillet@rakuten.com
Rakuten Group, Inc.

ABSTRACT

Position bias removal from ranking models learned using implicit feedback is extensively studied in the literature for classical web search. To this end, significant research efforts are made on counterfactual LTR, which reduces bias using the probability of intervention, termed propensity scores. Estimating these propensity scores is not straightforward. Intervention harvesting [4] is one notable approach to estimate propensities from observational data for classical web search. However, only a few works [7, 11] exist for bias removal from an e-commerce product search. Specifically, harvesting methods have not been studied in this product search framework. Furthermore, while learning an unbiased LTR in this setting by applying different propensity scores, their influence on other elements, such as the loss function and the optimizer, are yet to be explored.

This work attempts to bridge these gaps by studying three essential elements of counterfactual LTR in a real-life large-scale e-commerce setting: harvesting-based propensity estimators, pairwise-additive LTR losses, and gradient-based optimizers. Specifically, we aim to analyze the predictive performance of counterfactual LTR under these three experimental axes. Moreover, there is no comprehensive study to analyze the influence of these different elements in counterfactual LTR in our setting.

Our experiments confirmed an underlying dependency on the optimizer-loss configuration.In particular, we observed that propensity correction changes the learning dynamics for accelerated gradient-based optimizers. Thus, counterfactual LTR benefits in tandem with non-accelerated gradient descent; otherwise, it degrades the performance in a real-life scenario.

CCS CONCEPTS

• **Information systems → Learning to rank**.

*Both authors contributed equally to this research.
†This research was done during an internship at the Rakuten Institute of Technology.

SIGIR eCom'22, July 15, 2022, Madrid, Spain

KEYWORDS

position bias, unbiased learning-to-rank, propensity estimation, industrial application

ACM Reference Format:
Parantapa Goswami, Andrés Hoyos-Idrobo, Théo Deschamps, and Nicolas Feuillet. 2022. From Theory to Practice: Testing Harvesting-Based Propensity Estimators on Counterfactual Learning-To-Rank from Implicit Feedback for E-commerce Search . In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'22).* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Using machine learning to build ranking models, collectively known as Learning-to-Rank (LTR), is the de-facto practice in search paradigm. These models typically rely on expert annotated relevance judgments or implicit user feedback for training and evaluation. As opposed to human evaluated judgments, implicit user feedback is abundantly available and reflects meaningful user preferences, making it suitable for the LTR model fitting.

This work concentrates on a more specific use case, the e-commerce product search. For this type of search, the relevance of a product is not limited only to matching the content description with the issued query. It encompasses other factors like the utility of the product and price [24]. Moreover, user feedback in e-commerce product search goes beyond clicks, as signals like add-to-cart, bookmark, and purchase are also available, thus making implicit feedback a more cost-effective option.

However, the nature of user interactions is inherently biased [15], making straightforward application of implicit feedback difficult. One such significant bias is the position bias, where a user tends to examine and thus interact with higher-ranked products more than the lower-ranked ones. Consequently, she does not click potentially relevant documents on lower-ranking positions. The seminal study by Joachims et. al. [16] proposed unbiased LTR, a counterfactual learning approach to reduce biases systematically. Under this framework, we divide a ranking loss function by the probability of examining a document at a clicked position, i.e., propensity score; however, this requires proper knowledge of such probabilities. Hence, accurate propensity estimation is paramount for an effective unbiased LTR.

Online interventions are still the gold standard to estimate exposure-at-position propensities, specifically ranking result randomization [15]. However, in practical scenarios, these interventions are disruptive

and significantly hamper the search experience [27], leading to additional unseen costs. Therefore, current research in counterfactual LTR concentrates on using causal models of users' implicit feedback to leverage observational data and bypass these interventions. Notably, these methods include propensity estimation [4, 7, 27] and designing custom ranker's architecture [5, 28].

This work focuses on using propensity estimation methods in counterfactual LTR. Specifically, we use interventional harvesting-based propensity estimators [4]. This class of estimators uses historical user click-logs where various rankers may rank the same query-document pair at different positions. They emulate the effect of ranking result randomization. This enables us to compute the propensities offline.

We go from theory to practice, testing intervention harvesting estimators with counterfactual LTR on a large-scale real-life e-commerce query log obtained from the Rakuten product search engine. We aim to answer the following questions: *i)* How do harvesting-based propensity estimators behave in an e-commerce setting? *ii)* How do different loss functions and optimizers influence the ranking performance of counterfactual LTR methods corrected using harvesting-based propensity estimations?

Our contribution is twofold:

- We tested interventional harvesting-based propensity estimators on a real-life e-commerce dataset and analyzed their limitations in this setting.
- We empirically study the relationship between the different elements in unbiased LTR, such as the loss function, the optimizers, and the estimated propensities.

To the best of our knowledge, this is the first work to do so. These aspects of counterfactual LTR are well studied for the classical search (Sec. 3). The same is missing for e-commerce product search.

2 RAKUTEN ICHIBA USE CASE

Rakuten Ichiba is the largest e-commerce marketplace in Japan hosting nearly 56 thousand merchants selling around 360 million items[1]. The product search engine of Rakuten Ichiba, henceforth referred to as Rakuten, receives around 130 million queries per day and has to retrieve relevant items from the catalog containing these 360 million items. Frequent A/B testings to validate ranking models or similar online interventions on such a large-scale product search engine result in the potential risk of loss of revenue. Thus, we deliberately limited the context of this work to offline propensity estimators to avoid performing online interventions.

We focus on the desktop version of the Rakuten website, where the search engine results pages (SERPs) display results in a 2-dimensional grid. Under the default filters setting, each result page presents 45 items with five items per row, leading to 9×5 SERPs. Only the first page for each query is logged in the dataset. Fig. 1 shows a typical SERP of Rakuten[2].

2.1 Rakuten LTR dataset

We performed all the experiments using search logs collected from the Rakuten product search engine. Our dataset contains 2% of users' queries, sampled randomly, during the period of April 3rd, 2021 to April 19th, 2021. Products in a result page can typically be sorted using various options, such as best match, descending price or ascending price, user review scores, new arrival etc. For this paper, following [7], we log results sorted according to the best match. The final dataset contains 556 252 queries with 40.322 average number of result products per query. Normally each query contains 45 result items. However, queries for very specific products, or queries with specific filters activated (e.g. color, genre) result in fewer items in the SERP. Table 1 details the dataset.

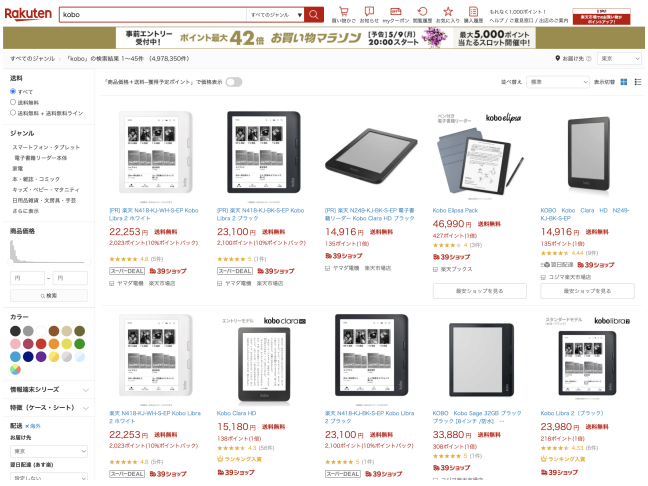

Figure 1: A typical web SERP of Rakuten Ichiba

The feature set has 70 features, including query, product, and query-product dependent features. The query features rely on its structure, such as its length. The product features include information on the product's description and previous user exchanges with the product. Finally, the query-product features enclose mainly similarity-based information such as SOLR[3] similarity fields[4]. Also, the data collection pipeline records three types of user interactions with the search results, clicks, add-to-carts, and purchases.

Fig. 2 shows the frequency of clicks and purchases for different rank positions. We fit an power-law model on the top 45 positions, $f(x) = a\ (x^{\sigma}) + b$, where a denotes the amplitude, b is the offset, and σ is the decay constant. Thus, the power-law model accurately depicts the observed popularity ranking. In particular, we have for clicks and purchases $\sigma = -0.89$, $a = 8$k and $b = -150$ for purchases, and $a = 48$k and $b = 236$ for clicks. This decay shape, in particular, the lack of oscillations every five items, may suggest that users examine items in a 1-dimensional list or sequential reading, from left to right and top to bottom. Nevertheless, items are displayed in a 2-dimensional grid. We confirm this intuition in Sec. 6.

2.2 Limitations

Business constraints compel us to deploy bias removal solutions with minimal disruption to the current functional pipelines—this constrains our exploration of more complex ranker architectures, such as the Dual Learning algorithm [6]. Consequently, we adopted

[1] https://rakuten.today/blog-ja/rakuten-ichiba-25years-2022-j.html?lang=ja
[2] Search date: May 9, 2022. SERP URL: https://search.rakuten.co.jp/search/mall/kobo/
[3] https://solr.apache.org/
[4] https://solr.apache.org/guide/8_4/other-schema-elements.html

Table 1: LTR dataset description

Number of samples (queries)	556 252
Average number of products per query	40.322
Number of features	70
Click rate (% clicks among impressions)	5.553%
Conversion rate (% purchases among impressions)	0.193%

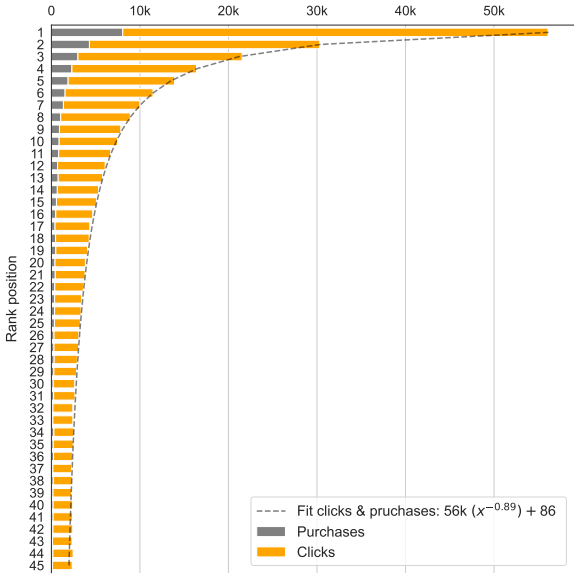

Figure 2: Frequency of the click and purchase feedback signals along different rank positions in the LTR dataset.

a simple unbiased LTR method, counterfactual LTR [16]. Also, we relied on linear rankers to limit the experimental design and used pairwise additive ranking losses as learning objectives. These ranking losses are ubiquitous in LTR as they include two staples of the field: RankNet and SVM-rank. Therefore, our problem boils down to estimating the propensities. Furthermore, since Rakuten Ichiba is a real-time product search engine, performing online interventions such as result randomization is not affordable, motivating us to turn towards offline propensity estimation techniques.

3 RELATED WORK

Two main strategies exist to avoid randomized experiments to assess exposure-at-position propensities in e-commerce. One is to tailor the architecture of the ranker [5, 28], and another one is to build propensity estimators [4, 7, 27]. Both methods impose a causal model of click generation to bypass interventions. However, we aim to use agnostic strategies to ranker's architecture; we rely on linear rankers. Hence, we use harvesting-based propensity estimations to set reweighting values in counterfactual LTR [17] (Sec. 4.2.2).

Propensity estimators. [4] introduced various estimators of exposure-at-position propensities from historical click-logs. The cornerstone of these methods is the interventional sets, a set of logs where two or more rankers rank the same query-document pair at different positions. They tested the performance of these estimators in classical search data. In parallel, [7] proposed a similar estimator to harvesting-based estimators in e-commerce settings. However, the performance of harvesting-based estimators in real-life large-scale e-commerce needs to be explored.

Regarding actual e-commerce data, [7] used 40k queries, whereas we relied on approximately 500k in our experiments; this is one order of magnitude more queries than previous studies.

Pairwise-additive LTR losses. [2] tested various counterfactual LTR with pairwise-additive losses. In particular, they introduced an SVM rank variant that optimizes Discounted Cumulative Gain (DCG), SVM-Rank DCG (Sec.4.2.1). They showed that this ranker outperforms other linear rankers in terms of DCG on semi-simulated data. To our knowledge, there are no comparisons of these LTR losses in counterfactual LTR in real large-scale e-commerce data.

Optimizers. [12] compared various solvers on semi-simulated data: Stochastic Gradient Descent (SGD), and some of it variants, ADAM and AdaGrad. They observed that, in all tested cases, learning with SGD in counterfactual LTR improves over clicks. However, other accelerated variants like ADAM fail in some instances. As far as we know, there are no comparisons of different optimizers in counterfactual LTR on real large-scale e-commerce data.

4 BACKGROUND AND METHODOLOGIES

We consider the problem of diminishing the position bias in LTR from biased clicks in an e-commerce environment. We first introduce the causal model of users' clicks. Then, we introduce LTR with pairwise additive metrics and its different learning signals. Finally, we present the harvesting-based propensity estimators.

We denote sets as calligraphic, e.g., $\mathcal{D}_q$, and vectors in bold, e.g., $\mathbf{p}$. We use $[K] = \{1, \ldots, K\}$ to denote a list of K elements. Also, we use query-document (q, d) to index feature vectors to avoid cluttering by adding another variable.

4.1 Position-based propensity model

The position-based model (PBM) simulates users' click behavior by assuming that a click only depends on the displayed position and relevance of the document. Thus, a user clicks on a given document d if she observes it at position k and considers it relevant to the issued query q. The PBM is as follows:

$$P\left[c^k(d) = 1 \mid q\right] \propto P\left[e^k(d) = 1\right] \cdot \text{rel}(q, d) = \mathbf{p}_k \cdot \text{rel}(q, d), \quad (1)$$

where $\text{rel}(q, d)$ is the relevance of item d to query q, $c^k(d) \in [0, 1]$ and $e^k(d) \in [0, 1]$ correspond to the click and examination occurrences for document d at position k. In the PBM, the examination probability $\mathbf{p}_k$ is identical to the observation propensity [17], and it corresponds to the source of discrepancy between the clicks and relevance, our signal of interest.

4.2 LTR with pairwise additive losses

LTR aims to learn a ranker ϕ that predicts sortings from document-query features. We denote by $S_\phi(q)$ the sorting of all candidate

documents $\mathcal{D}_q$ for query $q \in \mathcal{Q}$. Thus, the output of $S_\phi(q)$ is the list of sorted/ranked documents in $\mathcal{D}_q$, $\mathbf{d} = S_\phi(q)$.

We rely on the class of pairwise additive losses as the objective function of the learning algorithm. Pairwise additive losses are ubiquitous in LTR literature and provide a pleasant formalism for click debiasing (using the PBM model).

4.2.1 Pairwise additive losses [2] . These metrics are expressed as:

$$\Delta_{\text{rel}}(\mathbf{d} \mid q) = \sum_{d \in \mathcal{D}_q} \lambda(\text{rank}(d \mid \mathbf{d})) \cdot \text{rel}(q, d), \tag{2}$$

where $\lambda(\cdot)$ is a monotonic weighting function. The class of linearly decomposable metrics contains many commonly used ranking metrics: average rank, DCG, precision at k, and RBP [20]. We consider negative values wherever necessary to make the notation consistent with risk minimization.

4.2.2 Debiasing Pairwise Additive Losses. Counterfactual LTR uses a generative user behavior model to reduce bias in clicks. Thus, it modifies the empirical loss by weighting each sample by its Inverse Propensity Score (IPS). The counterfactual loss is as follows

$$\Delta_{\text{IPS-click}}(\mathbf{d} \mid q) = \sum_{k=1}^{K} \frac{\lambda\left(\text{rank}(d \mid \mathbf{d})\right) \cdot c^k(d)}{\mathbf{p}_k}. \tag{3}$$

We have a propensity per query, $\mathbf{p}_k$, whenever we assume, one clicked item per query.

Eq. 3 is an unbiased estimate of the full-information loss, Eq. 2, if the propensities are correct and $\mathbf{p} > 0$ for all d that are relevant [17]. Thus, the learning algorithm is guaranteed to find an unbiased ranker for large enough training data. However, the theory indicates asymptotic convergence in the sample size.

4.3 Harvesting-based propensity estimators

We note that the performance of Counterfactual LTR requires knowing the truth position propensities $\mathbf{p} \in \mathbb{R}_+^K$. However, this requirement is often not met in practice, as performing interventions is frequently impossible for business-sensitive tasks due to their high costs. We can bypass interventions by simulating them via a click generating model, e.g., PBM. Harvesting-based propensity estimators [4] is one of such methods.

Harvesting-based estimators rely on click-logs from multiple rankers to synthesize interventions using co-occurrences statistics. Formally, these estimators use M rankers, $\{\phi_i\}_{i=1}^M$ with their respective click logs, i.e., query-document-click tuples, to emulate the effect of swapping pairs of ranking positions, (k, k'), in LTR systems. This work uses **Chain** and **All pairs** propensity estimators.

Harvesting-based estimators use the following assumption:

Assumption 1. *For all $i \in [M]$, the query distribution $P[Q]$ does not depend on the choice of the ranker ϕ_i.*

The previous assumption is realistic as the assignment of queries to rankers is randomized in A/B tests. The condition also implies:

$$\forall \phi_i : P[Q \mid \phi_i] = P[Q] \Rightarrow \forall q \in Q : P[\phi_i \mid q] = P[\phi_i].$$

Definition 4.1. The interventional set $\mathcal{S}_{k,k'}$ is the set of query-document pairs such that two rankers, ϕ and ϕ', put a document d at positions k and k', respectively. Then, the interventional set is:

$$\mathcal{S}_{k,k'} \coloneqq \{(q,d) \ : \text{rank}(d \mid S_\phi(q))) = k \wedge \text{rank}(d \mid S_{\phi'}(q)) = k', \\ \forall q \in Q,\ d \in \mathcal{D}_q\},$$

where $k \neq k' \in [K]$, and K is some fixed number of top positions for which propensity estimates are desired, e.g. $K = 10$.

The interventional set contains query-document pairs where the choice of the ranking function assigns at random a document d to position k or k'. Thus, the sets $\mathcal{S}_{k,k'}$ represent virtual swap interventions between ranks k and k'. Nevertheless, Definition. 4.1 does not require any query to occur multiple times.

Assumption. 1 ensures that the virtual swap is entirely randomized. However, the assignment is not uniform. We can reweight it to make it uniform. The weight $w(q, d, k)$ is the number of times a ranker ϕ_i ranks a document d at position k for $\mathbf{n}_i$ queries and $i \in [M]$, where $\mathbf{n}_i$ be the number of queries that ϕ_i processed,

$$w(q, d, k) \coloneqq \sum_{i=1}^{M} \mathbf{n}_i \, \mathbb{1}_{\left[\text{rank}(d|S_{\phi^i}(q)))=k\right]}.$$

All intervention harvesting estimators rely on the rate of clicks in position k (and in position k'):

$$\hat{\mathbf{c}}_k^{k,k'} = \sum_{i=1}^{M} \sum_{q \in Q^i} \sum_{d \in \mathcal{D}_q} \mathbb{1}_{[(q,d) \in \mathcal{S}_{k,k'}]} \mathbb{1}_{[\text{rank}(d|S_{\phi_i}(q))=k]} \frac{c(d)}{w(q,d,k)}.$$

4.3.1 Adjacent Chain estimator . The Adjacent Chain estimator (**Chain**) is a local estimator of position propensities, and it is an adaptation of the estimator used in [27]. It uses a chain of swaps between adjacent pairs ranked at positions k and $k-1$. To get the relative ratio between any two positions, we first compute the ratio of the number of clicks at position pairs $(k, k-1)$. Then, we multiply the ratios of adjacent positions for various k. In particular, we have the following propensity ratio for position pairs $(k, 1)$:

$$\frac{\hat{\mathbf{p}}_k}{\hat{\mathbf{p}}_1} = \frac{\hat{\mathbf{c}}_2^{1,2}}{\hat{\mathbf{c}}_1^{1,2}} \cdot \frac{\hat{\mathbf{c}}_3^{2,3}}{\hat{\mathbf{c}}_2^{2,3}} \cdot \ldots \cdot \frac{\hat{\mathbf{c}}_k^{k-1,k}}{\hat{\mathbf{c}}_{k-1}^{k-1,k}}. \tag{4}$$

This estimator is statistically consistent under mild conditions.

4.3.2 All Pairs estimator . The All Pairs estimator (**All pairs**) is a global estimator of position propensities, which leverages most of the information in the interventional sets $\mathcal{S}_{k,k'}$. It optimizes a weighted cross-entropy objective overall interventional sets. Thus, this objective requires computing the ratio of non-clicks $\neg\hat{\mathbf{c}}_k^{k,k'}$, which corresponds to replacing $c(d)$ for $1 - c(d)$ in Eq. 4.

Let $\mathbf{R} \in \mathbb{R}_+^{K \times K}$ be the matrix of expected relevances. For $k \neq k' \in [K]$:

$$\mathbf{R}_{k,k'} = \mathbb{E}_q \left[\sum_{d \in \mathcal{D}_q} \mathbb{1}_{[(q,d) \in \mathcal{S}_{k,k'}]} \text{rel}(q,\ d) \right]. \tag{5}$$

Let $\bar{\mathbf{R}}$ the normalized expected relevance, such that each $\mathbf{R}_{k,k'}$ is within the interval $[0, 1]$. Normalizing ensures that the contribution of each aggregated click-through sample is proportional to the size of its interventional set.

All pairs is the solution of the following training objective:

$$(\hat{\mathbf{p}}, \hat{\mathbf{R}}) \leftarrow \arg\max_{\mathbf{p},\mathbf{R}} \sum_{k \neq k' \in [M]} \hat{\mathbf{c}}_k^{k,k'} \log\left(\mathbf{p}_k \bar{\mathbf{R}}_{k,k'}\right) + \neg\hat{\mathbf{c}}_k^{k,k'} \log\left(1 - \mathbf{p}_k \bar{\mathbf{R}}_{k,k'}\right).$$

The optimization step in **All pairs** has $O(K^2)$ complexity, where K is usually small, e.g., $K < 100$. [7] proposed a similar propensity estimator method to **All pairs**. However, it does not require fitting a relevance model. Both estimators rely on equivalent modeling assumptions, and their difference in performance remains to be defined; thus, we selected to investigate estimators based on interventional sets.

4.4 Intervention harvesting for Rakuten

The primary concept of harvesting swap interventions exploits the use of search query log data collected during an A/B test. The data contains queries served by five different rankers deployed in conjunction during the A/B test, where each query is sent to one of the randomly selected rankers, thus satisfying Assumption 1. Around 22.5% of the queries were sent to different variants, each receiving around 4.5% of the queries, collecting 1 249 308 queries to estimate propensities through intervention harvesting.

The implementation of intervention harvesting [4] is different for Rakuten in two aspects. Firstly, even for identical user queries, the item-query features change over time based on numerous factors such as sales or offers. Secondly, the item catalog, the pool of items, is also constantly evolving. Therefore, the candidate set of items associated with a query varies over time. As a result, one ranker might yield two different rankings for the same query. Accordingly, we can relax the definition of interventional set (Definition 4.1) by not requiring that two different rankers ϕ and ϕ' rank a document d at positions k and k', respectively. Hence, the interventional set $\mathcal{S}_{k,k'}$ is the set of query-document pairs such that there exists one ranker Φ that ranks a document d at positions k and k'.

We impose an additional restriction to determine whether two queries are identical, ensuring that the item pool has not changed significantly between the two queries. Apart from an exact string match, we considered two queries the same if they share at least 90% items in the retrieved set. As mentioned above, the dataset logs only the first SERP of a query containing 45 items. Thus, two queries are identical if the query strings are an exact match and the logged SERP contains at least 42 common items.

5 EXPERIMENTAL SETUP

We conducted extensive experiments to study the effect of position bias correction on different perspectives of learning-to-rank models for Ichiba product search. Through the results of these experiments, we aim to answer the following research questions:

- **RQ1** How do harvesting-based propensity estimators behave in an e-commerce setting?
- **RQ2** How do different loss functions and optimizers influence the ranking performance of counterfactual LTR methods corrected using harvesting-based propensity estimations?

5.1 Training and evaluation

Reliable relevance judgment strongly affects the potency of the learned LTR model. However, the notion of relevance in e-commerce product search differs from a standard web search. For classical web search, the results are judged relevant by human annotators if the textual content in the result corresponds to the intent of the issued query. On the contrary, the utility of a product goes beyond the textual content and involves a collection of different parameters, such as its brand, price, or user rating.

A typical e-commerce search log contains different levels of user interactions which serve as surrogates of the relevance of products against an issued query [24]. As mentioned in Sec. 2.1, Ichiba search logs have three feedback signals, namely clicks, added-to-carts, and purchases. After a thorough study, Santu et. al. [24] observed that training with purchase signals presents the most robust model, followed by clicks. Thus, the purchase denotes overall user satisfaction and is a more robust proxy for relevancy. However, due to the sparsity of the purchase signals, it is advised to use clicks for training initially. Motivated by these results, we have used clicks to train the learning-to-rank models and purchases to validate them and measure their performances.

Thus, we used the dataset (Section 2.1) for training and evaluation, where we rely on clicks during training and purchases for evaluating the models. We evaluated using nDCG@10 [13], considering the highest 10 ranked items.

5.2 Methods to compare

We compared bias correction methods using different estimated propensities against no correction. In turn, all these models use four different loss functions and train using four different optimizers.

5.2.1 Propensity estimators. We aim to fit a ranker ϕ by minimizing the Counterfactual LTR loss, Eq. 3. We rely on various estimators of the position propensities, as follows:

- **Naive** It learns on user feedback signals ignoring any bias, i.e., clicks.It boils down to setting each position propensity to one, $\mathbf{p}_k = \mathbf{1}$ for $k \in [K]$, and it is our baseline.
- **REM [27]** It is a learning-based unbiased estimation of propensities that does not depend on result randomization by relying on the PBM model to avoid interventions. It alternates between fitting a ranker and an examination predictor. **REM** assumes that the joint optimum of the ranker and the propensity model produce unbiased estimates of propensity scores. We used a random forest classifier with 20 trees and Gini impurity as the splitting criterion in the maximization step. Each iteration uses the random forest of the previous iteration and refines it additively with the new batch of data. This method is the most straightforward data-driven propensity estimator; hence, we use it as a reference.
- **Chain [4]** The propensities are estimated by the adjacent chain estimator using the intervention sets (Sec. 4.3.1).
- **All pairs [4]** The propensities are estimated by the all pairs estimator using the intervention sets (Sec. 4.3.2)

5.2.2 Step: Row-wise quantization . As described in Sec. 2 and exhibited in Fig. 1, the default desktop result page in Ichiba presents the ranked items in a grid format with five items displayed per row. Intuitively one can assume that while scrolling through such a result page, users are exposed to five items in a row simultaneously, resulting in virtually the same examination probability for these

five items. Motivated by this assumption, we tested a quantized version of propensity scores specific to the Ichiba. We denoted this quantization as Step, it is calculated as follows:

$$\hat{\mathbf{p}}_k^{\text{step}} = \sum_{i=0}^{8} \frac{\mathbb{1}_{[k \in \mathcal{K}_i]}}{|\mathcal{K}_i|} \sum_{j \in \mathcal{K}_i} \hat{\mathbf{p}}_j, \tag{6}$$

where $\mathcal{K}_i := \{(5\,i), \ldots, (5\,i+5)\}$ for $i \in \{0, \ldots, 8\}$ is the set of five items for row i. The mean aggregation of row-wise propensities in Eq. 6 is a design choice, and it could be replaced by another statistical aggregate, e.g., median. We studied the Step version along with the Raw version of **REM**, **Chain**, and **All pairs**.

5.2.3 Rankers and learning. In this study, we are not evaluating the capacity of the ranker, i.e., finding the ranker with the best predictive performance. Instead, we aim to assess the influence of data-driven propensity estimation methods in ranking. Thus, we use a linear ranker ϕ (Sec. 4.2) to score the documents. The linear model cancels out the effect of the representation power or capacity, it leads to fast inference/prediction times, and it is widely used in LTR [19]. Also, it constitutes the original definition of SVM-Rank, a cornerstone in LTR research.

We use the following three pairwise-additive loss functions:

SVM-Rank [14] It is a linear classifier that computes the hinge loss between candidate pairs. It also uses the ℓ_2 regularization term to avoid overfitting.

SVM-Rank DCG [2] Modifies the SVM-Rank and constitutes a creates a relaxation of the DCG loss (Sec. 4.2.1).

RankNet [8] It computes the logistic loss for candidate pairs.

Also, we investigate the impact of the optimizer on the prediction performance of Counterfactual LTR with different propensity estimators. We consider four commonly used optimization methods: Regular SGD [23], AdaGrad [10], RMSProp [26], and ADAM [18].

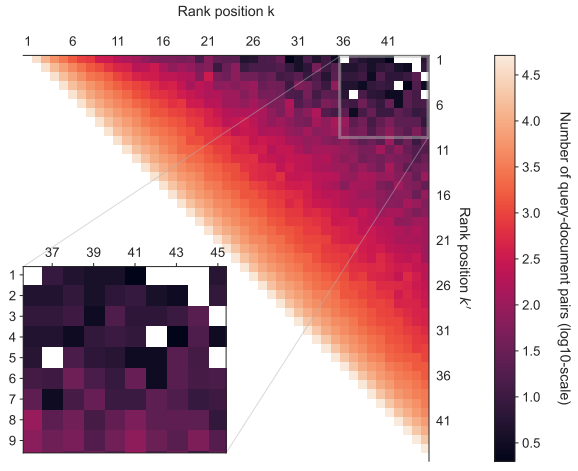

Figure 3: The size of interventional sets. For a given query-document pair, the frequency of ranking a document on top using a ranker ϕ and the bottom using a ranker ϕ' is low. White cells denote zero values.

5.3 Implementation

We use Pytorch [21] as the training framework for all the experiments. PytorchLTR [12] library implements various LTR loss functions based on Pytorch, including those tested in this work. We implemented the bias correction version of all the loss functions using the general PytorchLTR framework. Also, we employed the Pytorch implementation of the optimizers with their default parameters and set learning rates in $\{0.0075, 0.01, 0.05, 0.075, 0.1, 0.5\}$. Models are trained with mini-batches with a batch size of 64 queries.

We computed the interventional sets using PySpark[5].

6 RESULTS

6.1 Interventional sets for Rakuten search

It is required to compute the interventional set to calculate position propensity estimates with **Chain** or **All pairs**. Thus, we start by exploring the interventional sets obtained for Rakuten search logs.

Fig. 3 depicts the logarithm of the size of interventional set, $\log_{10}\left(\left|\mathcal{S}_{k,k'}\right|\right)$ for position ranks $k \neq k' \in [45]$ Due to symmetry, we show only the upper triangular part of the matrix, where white cells correspond to zero values. We observe a low frequency of swapping positions $k \leq 10$ and $k' \geq 35$. Specifically, there are instances where the intervention set $\mathcal{S}_{k,k'}$ is zero for $k \geq 35$. In other words, two rankers rarely position the same query-document pair at contrasting rankings positions, higher and lower, respectively. We see that swapping the top-one position with the bottom ten positions has four empty sets. Hence, the results for contrasting rankings are unreliable.

Even if high-contrast rankings are unreliable, we kept all position values in our experiments because these forty-five positions are essential for business, because it corresponds to our scenario, i.e. Rakuten Ichiba 9×5 grid display. Also, setting a threshold to the position value will hinder the out-of-the-box application of propensity estimates.

[5] https://spark.apache.org/docs/latest/api/python/

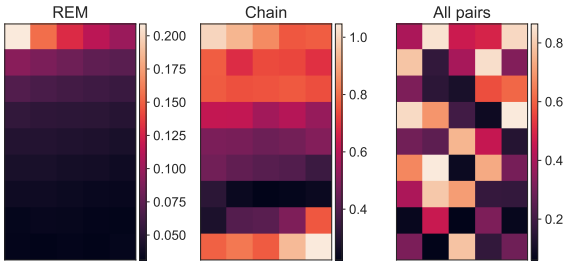

Figure 4: Propensities calculated using various data-driven estimators in a 9×5 grid display, following the layout of a typical Rakuten SERP. REM (*left*) produces position propensities according to a sequential reading assumption. Chain (*middle*) gets similar values row-wise with a trend to decrease from top to bottom. All pairs (*right*) leads to noisy propensities with a mild trend to decrease from top to bottom.

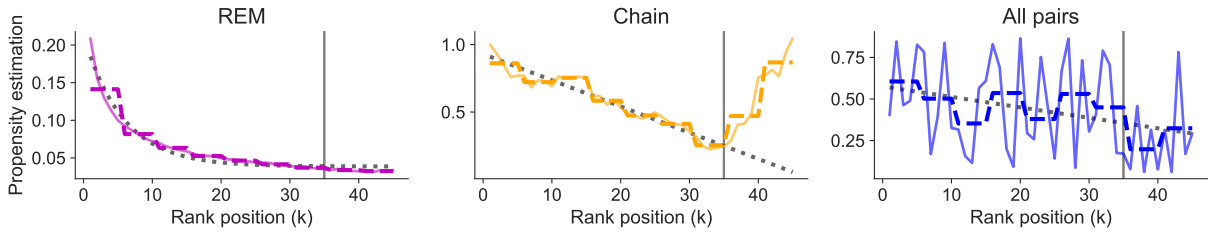

Figure 5: Position propensity estimated for Rakuten website using various estimation methods. The solid line is the direct estimation of propensities (Raw). The dashed line is the row-quantization of propensities (Step). The dotted gray line is the fitted decay model: exponential for REM, linear for Chain and All pairs. The vertical black line denotes the rank position 35.

6.2 RQ1 Estimating propensities for Rakuten

We computed several data-driven propensity estimations on Rakuten search logs and explored their behavior (Sec. 4.4).

Fig. 5 shows the plots of the estimated propensities using data-driven approaches and their row-based quantization (step). **REM** exhibits monotonically decreasing propensities due to the construction of the model. **REM** obeys the following exponential decay model, $\hat{p}(\text{rank}) = 0.17 \exp(-\text{rank}/5.73) + 0.03$, in contrast to the power-law decay in the observed data (Sec. 2.1). The propensities estimated with the **Chain** method display an overall linear decreasing trend up to the rank 35, $\hat{p}(\text{rank}) = 0.93 - 0.02\,\text{rank}$. After this rank, the **Chain** method is likely contaminated due to the number of clicks for consecutive rankings resembles, leading to values closer to one. Regarding **All pairs**, it produces noisy estimates due to unreliable interventional sets for high contrast rankings (Fig. 3). However, we still observe a decreasing linear trend, $\hat{p}(\text{rank}) = 0.57 - 0.006\,\text{rank}$.

Fig. 4 shows the grid display of the estimated propensities. The grid display is an array of nine rows and five elements/positions per row, following the layout of a typical Rakuten SERP. As said before, **All pairs** return noisy propensity estimates with a tendency to decrease values for the bottom rows. **All pairs** solves in principle a similar problem to **REM**; they both aim to find the propensity and relevance models, which is a more challenging endeavor compared to just estimating propensities, as is the case for **Chain**. Still, **All pairs** does not impose smoothness constraints on the solution, which explains, in part, this noisy behavior. On the other hand, **REM** produces less noisy propensities.

Nonetheless, in the central demography of Rakuten, Japanese people may have distinct browsing habits, e.g., observing documents from right to left. However, the **Chain** estimator gives higher values to the top row and lower values to the bottom rows. In particular, we observe decreasing row-wise propensity values up to row seven, where the estimated clicks are unreliable due to the lack of interventional sets. Nevertheless, there is not much difference between the values of the same row. These results contrast the sequential reading constraint of the standard **REM**. Regarding the **Chain** method, it seems oblivious to the display type, e.g., list or grid, because of its co-counting nature, and this result reassures our assumption of row-wise examination (Sec. 5.2.2). Mainly, the **Chain** method produces similar propensities to the ones reported in [11] for desktop.

6.3 RQ2 Performance of unbiased LTR

We assess the effectiveness of the position propensities estimated in the previous experiment to correct for position bias in pairwise losses-based LTR predictors, Eq. 3. Regardless, counterfactual LTR is asymptotically unbiased only for the correct propensity scores. Thus, we explore the effect of these empirical estimations of propensity scores on large-scale but finite e-commerce data, a non-asymptotic case. Additionally, we evaluated the impact of propensity estimates on the convergence of the optimizer to a good ranker.

Table. 2 presents the performance in terms of nDCG@10, of different propensity estimation methods and solvers. We assess statistical significance with a paired t-test with $p < 0.0001$, which considers multiple comparison tests. Bold denotes the maximum row-wise, and Blue denotes the maximum across losses for the same solver. Statistically significantly lower and higher nDCG@10 than **Naive** is denoted with $\triangledown$ and $\triangle$ respectively. Regarding losses, the SVM-Rank DCG is consistently the best performing model for all optimizers. In contrast, both SVM-Rank and RankNet are not robust to scaling.

The **Chain** estimator improves prediction when used with SGD and Step propensities. On the other hand, its Raw version only leads to significant positive effects for the RankNet model. For **All pairs**, its Raw version only improves performance in conjunction with the SGD solver.

For **REM**, its Raw version only improves: (SVM-Rank DCG, AdaGrad) and (RankNet, ADAM). These results indicate that the list assumption of the standard **REM** estimator may not be valid. However, its Step version also underperforms, suggesting a smoother decay, as is the case for **Chain** (Fig. 5).

Fig. 6 portrays the variability across all reweighting methods, i.e., row-wise variability in Table 2. This variability is crucial to understanding the significance levels depicted in Table 2 as some methods have low variance, and similar values can be significantly different under a t-test. The SVM-Rank DCG model displays the lowest variance and the best predictive performance for all solvers. Thus, the relaxation of the DCG loss is less sensitive to scale, making it the most robust among all tested losses. Meanwhile, the SVM-Rank model has enormous variability, and RankNet displays an intermediate behavior. These results suggest that the SVM-Rank and RankNet models require additional regularization to control the stability of the generalization error.

The SGD optimizer produces stable solutions for all propensity correction methods. However, we do not observe the same behavior

Table 2: Performance nDCG@10 for various propensity estimation methods and solvers. Bold denotes the maximum row-wise, and Blue denotes the maximum across losses for the same solver. Statistically significantly lower and higher nDCG@10 compared to Naive is denoted with $\triangledown$ and $\triangle$ respectively ($p < 0.0001$)

Solver	Loss	Naive	REM		All pairs		Chain	
			Raw	Step	Raw	Step	Raw	Step
AdaGrad	SVM-Rank DCG	0.4272	**0.4282**$^{\triangle}$	0.4270	0.4273	0.4252$^{\triangledown}$	0.4260$^{\triangledown}$	0.4272
	SVM-Rank	**0.4258**	0.3656$^{\triangledown}$	0.4191$^{\triangledown}$	0.4114$^{\triangledown}$	0.3205$^{\triangledown}$	0.3744$^{\triangledown}$	0.3192$^{\triangledown}$
	RankNet	0.4242	0.4149$^{\triangledown}$	0.4181$^{\triangledown}$	0.4228$^{\triangledown}$	**0.4248**	0.4233	0.4227$^{\triangledown}$
ADAM	SVM-Rank DCG	0.4252	0.4157$^{\triangledown}$	0.4239$^{\triangledown}$	0.4239$^{\triangledown}$	0.4248	**0.4255**	0.4244$^{\triangledown}$
	SVM-Rank	**0.4225**	0.3465$^{\triangledown}$	0.3989$^{\triangledown}$	0.3500$^{\triangledown}$	0.3502$^{\triangledown}$	0.3507$^{\triangledown}$	0.3505$^{\triangledown}$
	RankNet	0.4014	**0.4112**$^{\triangle}$	0.3439$^{\triangledown}$	0.3759	0.3502$^{\triangledown}$	0.3981$^{\triangledown}$	0.3501$^{\triangledown}$
RMSProp	SVM-Rank DCG	**0.4268**	0.4224$^{\triangledown}$	0.4239$^{\triangledown}$	0.4254$^{\triangledown}$	0.4248$^{\triangledown}$	0.4257$^{\triangledown}$	0.4265
	SVM-Rank	**0.4091**	0.3456$^{\triangledown}$	0.3989$^{\triangledown}$	0.2904$^{\triangledown}$	0.3498$^{\triangledown}$	0.3508$^{\triangledown}$	0.2904$^{\triangledown}$
	RankNet	0.4013	0.4099$^{\triangle}$	0.3434$^{\triangledown}$	0.3720$^{\triangledown}$	0.2902$^{\triangledown}$	**0.4209**$^{\triangle}$	0.3505$^{\triangledown}$
SGD	SVM-Rank DCG	0.4304	0.4278$^{\triangledown}$	0.4287$^{\triangledown}$	0.4305	0.4280$^{\triangledown}$	0.4281$^{\triangledown}$	**0.4306**
	SVM-Rank	0.4237	0.3937$^{\triangledown}$	0.3967$^{\triangledown}$	**0.4246**$^{\triangle}$	0.4233$^{\triangledown}$	0.4229$^{\triangledown}$	0.4244$^{\triangle}$
	RankNet	0.4230	0.3801$^{\triangledown}$	0.3966$^{\triangledown}$	0.4227	**0.4235**	0.4207$^{\triangledown}$	0.4228

in other solvers. In particular, there is a detriment in the performance of SVM-Rank for AdaGrad and RMSProp. [12] reported similar effects for SGD and showed that ADAM occasionally converges to a suboptimal solution for the Counterfactual LTR problem. We can attribute this convergence behavior to suboptimal solutions of exponential moving averaging variants of SGD like RMSProp and ADAM [22]. However, AdaGrad is not one of these variants and still has performance drops. The main difference between SGD and other solvers is that the base SGD has no memory of previous iterations, i.e., we set the momentum parameter to zero. Thus, the propensity reweighting affects the learning dynamics of the optimizer with a recollection of previous iterates.

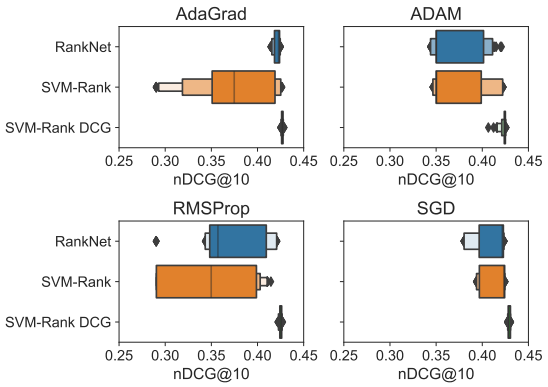

Figure 6: Variability of learning models across reweighting correction methods. SGD presents a minor variance and higher prediction accuracy than other solvers.

7 CONCLUSION

We conducted several counterfactual Learning-To-Rank (LTR) experiments on data from Rakuten Ichiba marketplace. Notably, we focused on the position-bias problem and left other sources of biases for future work, e.g., popularity [1], trust [3], and exposure [25]. We computed examination-at-position propensity estimates using various propensity estimation methods, **REM**, **Chain**, and **All pairs**, where the latter two are interventional harvesting-based estimators.

Regarding the propensity estimation (**RQ1**), we noticed that interventional harvesting-based methods are unreliable for high-contrast rankings due to data scarcity in these positions. Thus, these methods demand additional modeling information about the generative process. Notably, **All pairs** requires additional spatial smoothness constraints. Moreover, **REM** produces a sequential-like reading of documents, which does not lead to any improvement in prediction accuracy.

While assessing the influence of loss functions and optimizers on the performance of counterfactual LTR (**RQ2**), we did not see a clear pattern of the scaling correction across pairwise-additive losses. This lack of pattern hints at an implicit dependency on the optimizer-model configuration. Also, it empirically indicates that propensity correction changes the learning dynamics for accelerated or memory-based gradient-based optimizers; in most cases, these solvers negatively affect the predictive metric. Accordingly, SGD has robust predictive behavior on real data, which confirms what [12] reported on semi-synthetic data. In particular, SGD impacts the SVM-Rank positively when combined with **All pairs** or **Chain** with row-wise quantization, without affecting other rankers. Nevertheless, we saw that the effect of data-driven propensity estimators is challenging to assess in industrial settings, even for linear models. Still, we believe that the simplicity of linear models should provide a better insight into debiasing LTR losses than more complicated predictors or losses.

We consider as future work extending our experiments to list-wise losses and some state-of-the-art methods like LambdaMART [9] and non-linear rankers. We also consider conducting an in-depth exploration of the effect of propensity scaling on the learning dynamics of accelerated SGD methods.

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *ACM RecSys*. 42–46.

[2] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A general framework for counterfactual learning-to-rank. In *ACM SIGIR*. 5–14.

[3] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019. Addressing trust bias for unbiased learning-to-rank. In *WWW*. 4–14.

[4] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating position bias without intrusive interventions. In *ACM WSDM*. 474–482.

[5] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *ACM SIGIR*. 385–394.

[6] Qingyao Ai, Jiaxin Mao, Yiqun Liu, and W. Bruce Croft. 2018. Unbiased Learning to Rank: Theory and Practice. In *ACM CIKM* (Torino, Italy) *(CIKM '18)*. ACM, New York, NY, USA, 2305–2306. https://doi.org/10.1145/3269206.3274274

[7] Grigor Aslanyan and Utkarsh Porwal. 2019. Position Bias Estimation for Unbiased Learning-to-Rank in eCommerce Search. In *SPIRE*. Springer, 47–64.

[8] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *ICML*. 89–96.

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

[10] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12, 7 (2011).

[11] Ruocheng Guo, Xiaoting Zhao, Adam Henderson, Liangjie Hong, and Huan Liu. 2020. Debiasing grid-based product search in e-commerce. In *SIGKDD*. 2852–2860.

[12] Rolf Jagerman and Maarten de Rijke. 2020. Accelerated Convergence for Counterfactual Learning to Rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3397271.3401069

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[14] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *SIGKDD*. 133–142.

[15] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately Interpreting Clickthrough Data as Implicit Feedback. In *ACM SIGIR*. 154–161.

[16] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. ACM, 781–789. https://doi.org/10.1145/3018661.3018699

[17] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *ACM WSDM*. 781–789.

[18] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[19] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[20] Alistair Moffat and Justin Zobel. 2008. Rank-biased precision for measurement of retrieval effectiveness. *ACM TOIS* 27, 1 (2008), 1–27.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[22] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2019. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237* (2019).

[23] Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics* (1951), 400–407.

[24] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and Chengxiang Zhai. 2017. On application of learning to rank for e-commerce search. In *ACM SIGIR*. 475–484.

[25] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *SIGKDD*. 2219–2228.

[26] Tijmen Tieleman, Geoffrey Hinton, et al. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[27] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *ACM WSDM*. 610–618.

[28] Honglei Zhuang, Zhen Qin, Xuanhui Wang, Michael Bendersky, Xinyu Qian, Po Hu, and Dan Chary Chen. 2021. Cross-positional attention for debiasing clicks. In *WWW*. 788–797.