Towards Sequential Counterfactual Learning to Rank

Tesi Xiao
tesixiao@amazon.com
Amazon
Palo Alto, CA, USA

Branislav Kveton
bkveton@amazon.com
Amazon Web Services
Santa Clara, CA, USA

Sumeet Katariya
katsumee@amazon.com
Amazon
Santa Clara, CA, USA

Tanmay Gangwani
gangwt@amazon.com
Amazon
Palo Alto, CA, USA

Anshuka Rangi
anshuka@amazon.com
Amazon
Palo Alto, CA, USA

ABSTRACT

Counterfactual evaluation plays a crucial role in learning-to-rank problems, as it addresses the discrepancy between the data logging policy and the policy being evaluated, due to the presence of presentation bias. Existing counterfactual methods, which are based on the empirical risk minimization framework, aim to evaluate the ability of a ranking policy to produce optimal results for a single query using implicit feedback from logged data. In real-world scenarios, however, users often reformulate their queries multiple times until they find what they are looking for and then provide a feedback signal. In such circumstances, current counterfactual approaches cannot assess a policy's effectiveness in delivering satisfactory results to the user over consecutive queries during a search session. Taking sequential search behavior into account, we propose the first counterfactual estimator for session ranking metrics under sequential position-based models and conduct preliminary experiments to shed light on further research in this direction.

CCS CONCEPTS

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

KEYWORDS

Information Retrieval, Counterfactual Learning to Rank

ACM Reference Format:
Tesi Xiao, Branislav Kveton, Sumeet Katariya, Tanmay Gangwani, and Anshuka Rangi. 2023. Towards Sequential Counterfactual Learning to Rank. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP '23), November 26–28, 2023, Beijing, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3624918.3625325

1 INTRODUCTION

Learning to rank (LTR) is a critical component of modern search systems and is used in many applications such as web-search, e-commerce, and recommender systems [21]. The LTR module is responsible for sorting items based on their relevance to a user's query. Apart from supervised learning using expensive human annotations, many LTR techniques rely on implicit feedback such as clicks, dwell-time, and conversion as the reward [1]. The objective of LTR is to improve the users' search experience. Modern LTR techniques [2, 11, 22, 33] consider not only lexical similarity but also semantic similarity, past queries and actions taken by the user, and other meta-data such as location, time, etc.

LTR can be classified as either *online* or *offline* [21]. Online LTR incrementally trains a model with streaming incoming data, using techniques similar to multi-armed bandits [16, 25, 34]. In contrast, offline LTR trains a model on a static dataset of user engagements [1, 15]. Offline LTR has its own challenges, the primary being the bias in training data. For example, users are more likely to examine the top positions of search engine results pages (SERPs) and favor top-ranked items over items listed below. Naively trained models on such datasets will not accurately capture user preferences and item relevances. *Counterfactual* LTR (C-LTR) techniques address the problem of learning from biased logged datasets by incorporating the propensity of obtaining a particular training example into the Empirical Risk Minimization (ERM) objective [15, 30].

C-LTR is a technique used for estimating the impact of moving items within a single search engine ranking. However, in practice, users typically perform a series of queries within a session until they find what they are looking for [6]. This sequence of queries reflects the user's intentions and dissatisfaction with the initial ranked results. By extending C-LTR over a sequence of queries, we can estimate the effect of moving items between queries. Refer to Figure 1 for an example. A user who searches for an affordable wireless headphone with a good sound quality may start with a broad search term such as "wireless headphone" and then gradually refine their query to "affordable wireless headphone" and "affordable wireless headphone with good sound quality" if the top-ranked results do not match their expectations. In this case, if considering purchases as the reward, conventional C-LTR techniques will only estimate the impact of re-ranking items for the last query with a purchase record. However, taking sequential behaviors into account, we can argue that moving the later purchased headphone to the top of the recommended list under the first query may enhance the user's search experience.

Motivated by the above problem, we focus on *sequential* counterfactual LTR (SC-LTR) in this work, which aims to estimate the effect of re-ranking all recommended items in a search session from the history of queries and user engagements. Specifically, SC-LTR does not fall within the domain of sequential recommendation. Unlike existing sequential recommendation techniques (e.g., [3, 31, 32]) that

SIGIR-AP '23, November 26–28, 2023, Beijing, China

ACM ISBN 979-8-4007-0408-6/23/11.
https://doi.org/10.1145/3624918.3625325

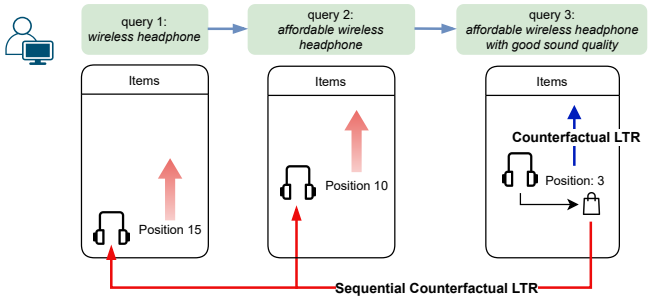

Figure 1: Counterfactual LTR improves single query rankings with user feedback, while Sequential Counterfactual LTR enhances search experience across queries.

aim to predict *future* user actions using past interactions, SC-LTR considers how to leverage *later* user actions to improve rankings for *previous* queries from offline data.

In this work, we propose SC-LTR and make several contributions. First, we formulate SC-LTR in a general setting and use metrics from the literature to evaluate the performance of SC-LTR algorithms. Second, we propose a sequential position-based model (sPBM) that simplifies the problem and enables the design of an unbiased counterfactual estimator. Third, we derive this estimator for SC-LTR. Finally, we conduct experiments that compare a sequential policy gradient (PG) method that uses our proposed estimator with vanilla PG and LTR algorithms.

2 PRELIMINARIES

In the sequel, we use bold symbols to represent vectors. Let $\mathbf{X} \in \mathcal{X}$ denote context and $\mathcal{X} \subseteq \mathbb{R}^d$ be the space of all contexts. The context $\mathbf{X}$ can include all the information from the query, items, and user. We denote the set of all documents by $\mathcal{D}$ and refer to the documents as items. For each context $\mathbf{X}$, we assume that the ranker is provided with a finite matched set of items denoted by $\mathcal{D}_\mathbf{X}$, which is a common practice in real-world ranking systems. Let $n_\mathbf{X} = |\mathcal{D}_\mathbf{X}|$ be the number of these items and $\mathbf{A} = (A_1, \dots, A_{n_\mathbf{X}}) \in S(\mathcal{D}_\mathbf{X})$ be a ranked list of $n_\mathbf{X}$ displayed items, where $S(\mathcal{D}_\mathbf{X})$ is the set of all permutations from $\mathcal{D}_\mathbf{X}$ and A_k is the k-th item in $\mathbf{A}$. Let $\mathbf{Y} = (Y_1, \dots, Y_{n_\mathbf{X}})$ be feedback on these items considered as reward signals. The random reward $\mathbf{Y}$ depends on the displayed list $\mathbf{A}$ and an underlying environment of user preferences. Let $R_\mathbf{X} = \sum_{k=1}^{n_\mathbf{X}} \lambda_k Y_k$ be the reward under context $\mathbf{X}$ in which the observed feedback at position k is weighted by λ_k.

Our goal is to evaluate and optimize ranking policies π, each of which is a categorical distribution over $S(\mathcal{D}_\mathbf{X})$ given context $\mathbf{X}$. A ranked list is generated according to policy π, which we denote by $\mathbf{A} \sim \pi(\cdot \mid \mathbf{X})$, as follows. The first item in the list is generated as $A_1 \sim \pi(\cdot \mid \mathbf{X})$, the second as $A_2 \sim \pi(\cdot \mid A_1, \mathbf{X})$, and the k-th as $A_k \sim \pi(\cdot \mid \mathbf{A}_{1:k-1}, \mathbf{X})$, where $\pi(A_k \mid \mathbf{A}_{1:k-1}, \mathbf{X})$ is the conditional probability that item A_k is displayed at the position k given a fixed list of all higher ranked items $\mathbf{A}_{1:k-1}$. The probability that list $\mathbf{A}$ is generated in context $\mathbf{X}$ is given by

$$\pi(\mathbf{A} \mid \mathbf{X}) = \prod_{k=1}^{n_\mathbf{X}} \pi(A_k \mid \mathbf{A}_{1:k-1}, \mathbf{X}).$$

Under this setting, the value of policy π for this query is defined as

$$V_{\text{query}}(\pi) = \mathbb{E}_\pi\left[R_\mathbf{X}\right] = \mathbb{E}_\pi\left[\sum_{k=1}^{n_\mathbf{X}} \lambda_k Y_k\right],$$

where the reward $R_\mathbf{X}$ is linear in user feedback $\mathbf{Y}$ that depends on π. By choosing λ_k accordingly, the above value function can represent the most common linear additive ranking metrics over reward $\mathbf{Y}$:

- Cumulative Gain (CG)@K: $\lambda_k = \mathbb{1}\{k \le K\}$
- Discounted CG (DCG)@K: $\lambda_k = \mathbb{1}\{k \le K\} / \log_2(k+1)$
- Rank-Biased Precision (RBP)@K: $\lambda_k = (1-q)q^{k-1}\mathbb{1}\{k \le K\}$, where q is the persistence parameter

It is important to emphasize that our objective is defined within the realm of user feedback-derived rewards, as opposed to the inherently ambiguous domain of relevance labels in conventional LTR [21].

2.1 Unbiased Evaluation via Inverse Propensity Scores

Let π_0 be the data logging policy that collects user feedback data. To indicate the variables collected under the data logging policy, we use a tilde, for example, π_0 recommends $\tilde{\mathbf{A}}$ given context $\mathbf{X}$ and observes $\tilde{\mathbf{Y}}$. A logged query-level dataset consisting of N queries indexed by i is defined as $\mathcal{D}_{\text{query}} = \{(\mathbf{X}^{(i)}, \tilde{\mathbf{A}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})\}_{i=1}^N$. We work on offline policy optimization problems, in which we want to find a better ranking policy π with larger expected reward per query from $\mathcal{D}_{\text{query}}$. Using the importance sampling trick, a common unbiased estimator of $V_{\text{query}}(\pi)$ using Inverse Propensity Scores (IPS) is

$$V_{\text{query}}(\pi) = \mathbb{E}_{\pi_0}\left[\frac{\pi(\mathbf{A} \mid \mathbf{X})}{\pi_0(\mathbf{A} \mid \mathbf{X})}\tilde{R}_\mathbf{X}\right]. \tag{1}$$

However, the above IPS estimator suffers from a large variance when the propensity ratio is large. Moreover, the data logging policy π_0 of real-world ranking systems is often unknown and can even vary over time, which makes the estimator difficult to use. Therefore, additional structural assumptions are proposed for off-policy evaluation of ranking policies [18].

2.2 Unbiased Evaluation via Position-Based Model

The position-based model (PBM) [28] is a model to handle the position bias where the probability of observing a reward (click, purchase, dwell time, etc.) from an item in the ranked list factors across its identity and position. Formally, given context $\mathbf{X}$, the reward Y_k observed from the item A_k at position k of list $\mathbf{A}$ is $Y_k = E_k U_{A_k}$, where E_k is a Bernoulli random variable indicating if position k is examined and U_a is a random variable characterizing the utility of item a. The examination of position k is sampled independently of all other variables as $E_k \sim \text{Ber}(p_k)$, where $\text{Ber}(p)$ is a Bernoulli distribution with mean p and p_k can be viewed as the probability of observing feedback at position k. The expected

utility of item a is $\theta_a = \mathbb{E}[U_a]$. Therefore, the expected reward of item a at position k is

$$\mathbb{E}[Y_k | A_k = a] = p_k \theta_a .$$

A good way of interpreting $\boldsymbol{\theta} = (\theta_a)_{a \in \mathcal{D}}$ is as user preferences. For example, when the reward is the click indicator, the utility of item a is sampled independently of all other variables as $U_a \sim \text{Ber}(\theta_a)$ where θ_a is the attraction probability of item a. Before the user interacts with the ranker, $(\theta_a)_{a \in \mathcal{D}}$ is drawn from some preference distribution, which we do not know. However, $\boldsymbol{\theta}$ and $\mathbf{X}$ are correlated given the intent of the user, and therefore policies that condition on $\mathbf{X}$ can recommend highly attractive items under unknown $\boldsymbol{\theta}$. An unbiased counterfactual estimator for $V_{\text{query}}(\pi)$ under the assumption of PBM is presented below.

$$\begin{aligned} V_{\text{query}}(\pi) &= \mathbb{E}_{\mathbf{A}, \tilde{\mathbf{A}}} \left[\sum_{k=1}^{n_{\mathbf{X}}} \lambda_k \sum_{\ell=1}^{n_{\mathbf{X}}} \frac{p_k}{p_\ell} \mathbb{1}\left\{A_k = \tilde{A}_\ell\right\} \tilde{Y}_\ell \right] \qquad (2) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{n_{\mathbf{X}}^{(i)}} \lambda_k \sum_{\ell=1}^{n_{\mathbf{X}}^{(i)}} \frac{p_k}{p_\ell} \mathbb{1}\left\{A_k^{(i)} = \tilde{A}_\ell^{(i)}\right\} \tilde{Y}_\ell^{(i)} = \hat{V}_{\text{query}}(\pi), \end{aligned}$$

where $\mathbf{A} \sim \pi(\cdot \mid \mathbf{X})$, $\tilde{\mathbf{A}} \sim \pi_0(\cdot \mid \mathbf{X})$, and $\mathbb{1}\{A_k = \tilde{A}_\ell\}$ is the indicator of the aforementioned event. In the expectation above, we only write randomness over $\mathbf{A}$ and $\tilde{\mathbf{A}}$ to reduce clutter.

This estimator is unbiased, as we prove as follows. The first observation is that the expectation of the total reward can be factored into the examination probabilities of all positions and attraction probabilities of items at those positions,

$$V(\pi) = \mathbb{E}_{\mathbf{A}} \left[\sum_{k=1}^{n_{\mathbf{X}}} \lambda_k Y_k \right] = \mathbb{E}_{\mathbf{A}} \left[\sum_{k=1}^{n_{\mathbf{X}}} \lambda_k p_k \theta_{A_k} \right] . \qquad (3)$$

Due to the structure of the position-based model, for lists $\mathbf{A}$ and $\tilde{\mathbf{A}}$, and position k in list $\mathbf{A}$, we have $p_k \theta_{A_k} = \sum_{\ell=1}^{n_{\mathbf{X}}} \frac{p_k}{p_\ell} \mathbb{1}\{A_k = \tilde{A}_\ell\} p_\ell \theta_{\tilde{A}_\ell}$. Given that this identity holds for any list $\tilde{\mathbf{A}}$, it also holds in expectation over $\tilde{\mathbf{A}} \sim \pi_0(\cdot \mid \mathbf{X})$. Then, we can complete the proof by applying it to (3). This proof is a straightforward extension of existing literature on C-LTR [1, 15].

The above estimator is slightly different from the standard position-based inverse propensity weighting estimator in C-LTR [1], which considers ground-truth relevance labels under the ERM framework and does not have an additional factor of p_k. However, if the objective is to maximize cumulative reward, i.e., $\lambda_k \equiv 1$, the above estimator becomes the standard position-based IPS estimator of DCG weighted by p_k, which has a clearer interpretation from our user behavior assumptions than traditional weights $1/\log_2(k+1)$.

3 SEQUENTIAL COUNTERFACTUAL LTR

Counterfactual evaluation has been focused on the offline evaluation of a search engine for serving the best results given a single query (Section 2). However, this query-level evaluation does not reflect a more realistic search scenario that takes into account the interaction between users and the search engine over multiple rounds and queries. In real-world scenarios, query reformulations can be numerous and persist until the user either finds what they look for or leaves frustrated [6, 12]. Therefore, session-level evaluation is also important because it provides a more realistic representation of how a user experiences a search engine and how it affects their overall search efficiency. Similar to the query-level counterfactual evaluation, we can define the session-level value function as follows.

Let $\{\mathbf{X}_1, \ldots, \mathbf{X}_H\}$ be a sequence of contexts which represents an H-round search session, $\mathbf{A}_s = (A_1, \ldots, A_{n_s})$ be a ranked list of n_s displayed items for context $\mathbf{X}_s$, $\mathbf{Y}_s = (Y_{s,1}, \ldots, Y_{s,n_s})$ be feedback on these items. The reward at round s, denoted by R_s, is calculated as the weighted sum of the observed feedback at position k using the weight $\lambda_{s,k}$. This is similar to the query-level reward. Under this setting, the session-level value of policy π is defined as

$$V_{\text{session}}(\pi) = \mathbb{E}_\pi \left[\sum_{s=1}^{H} R_s \right] = \mathbb{E}_\pi \left[\sum_{s=1}^{H} \sum_{k=1}^{n_s} \lambda_{s,k} Y_{s,k} \right],$$

Given this framework, session DCG (sDCG) [14] and session RBP (sRBP) [19] can be derived under different definitions of discount functions that can be factorized into query weight for round s and position weight for rank k:

- sDCG: $\lambda_{s,k} = \frac{1}{[1 + \log_{b_q}(1+s)][1 + \log_{b_r}(1+k)]}$. Here b_q, b_r are logarithm bases for queries and ranks respectively
- sRBP: $\lambda_{s,k} = \left(\frac{(1-b)q}{1-bq}\right)^{s-1} (bq)^{k-1}$. Here b is the balance parameter, which balances between reformulating queries and examining more documents. $p \in [0,1]$ is the persistence parameter defining the persistence of users in continuing search.

3.1 Unbiased Evaluation via Sequential PBM

In the sequential setting, a naive extension of (1) to the sequential setting will lead to an extremely high variance estimator since it would contain a product of propensities in each round in the denominator [20]. It also requires knowing the data logging policy. Both of these requirements make this extension impractical for most real-world applications. To avoid these issues, we leverage the idea of PBM and make the following assumptions on user behavior:

Sequential Position-Based Model (sPBM)

(1) The user interacts with the ranker for H rounds, which we call a session. The length of the session H can depend arbitrarily on θ. The context $\mathbf{X}_s$ for each round $s \in [H]$ includes all the information on query, matched items, user, and past interactions.
(2) In round $s \in [H]$, the recommender generates a ranked list $\mathbf{A}_s \sim \pi(\cdot \mid \mathbf{X}_s)$. The list is a permutation of items in $\mathcal{D}_{\mathbf{X}_s}$, which we abbreviate as $\mathcal{D}_s$. We also define a shorthand $n_s = |\mathcal{D}_s|$.
(3) The user interacts with the items as follows. The user feedback observed from item $A_{s,k}$ at position k in round s is

$$Y_{s,k} = E_{s,k} U_{A_{s,k}},$$

where the utility U_a of item a with mean θ_a is independent of all other variables, the examination of position k in round s is sampled independently of all other variables as $E_{s,k} \sim \text{Ber}(p_{s,k})$, and $p_{s,k}$ can be viewed as the probability of observing feedback at position k in round s.

(4) The user's intent remains the same and is not influenced by any recommendations they receive during the session. Specifically, the utility of each item (θ_a) does not change across rounds.

The sPBM is essentially a series of H chained PBMs, in which the utilities of items $\{\theta_a\}$ remain constant throughout the chain. Let π_0 be the data logging policy that collects user feedback data. Then, we will have a logged session-level dataset,

$$\mathcal{D}_{\text{session}} = \{(\mathbf{X}_1^{(i)}, \tilde{\mathbf{A}}_1^{(i)}, \tilde{\mathbf{Y}}_1^{(i)}, \ldots, \mathbf{X}_{H_i}^{(i)}, \tilde{\mathbf{A}}_{H_i}^{(i)}, \tilde{\mathbf{Y}}_{H_i}^{(i)})\}_{i=1}^{N},$$

that consists of N sessions indexed by i, where H_i is the length of the i-th session. At round j of session i, the context, displayed list, and reward are represented by $\mathbf{X}_j^{(i)}$, $\tilde{\mathbf{A}}_j^{(i)}$, and $\tilde{\mathbf{Y}}_j^{(i)}$, respectively. In all expectations below, we only write randomness over $\mathbf{A} = (\mathbf{A}_s)_{s\in[H]}$ and $\tilde{\mathbf{A}} = (\tilde{\mathbf{A}}_s)_{s\in[H]}$, which makes the change-of-measure argument easier to follow. Below, we derive an unbiased estimator of $V_{\text{session}}(\pi)$ under sPBM.

Theorem 3.1. *Given data collected using the policy π_0, the counterfactual estimator for the value of another policy π, which we name as session counterfactual DCG (SC-DCG), is given by*

$$V_{\text{session}}(\pi) = \mathbb{E}_{\mathbf{A},\tilde{\mathbf{A}}}\left[\sum_{s,z=1}^{H}\sum_{k,\ell=1}^{n_s,n_z} \frac{\lambda_{s,k}p_{s,k}}{p_{z,\ell}} \frac{\mathbb{1}\{A_{s,k} = \tilde{A}_{z,\ell}\}}{C(A_{s,k})} \tilde{Y}_{z,\ell}\right] \tag{4}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{s,z=1}^{H_i}\sum_{k,\ell=1}^{n_s^{(i)},n_z^{(i)}} \frac{\lambda_{s,k}p_{s,k}}{p_{z,\ell}} \frac{\mathbb{1}\left\{A_{s,k}^{(i)} = \tilde{A}_{z,\ell}^{(i)}\right\}}{C(A_{s,k}^{(i)})} \tilde{Y}_{z,\ell}^{(i)} = \hat{V}_{\text{session}}(\pi),$$

where $C(a) = \sum_{s=1}^{H} \mathbb{1}\{a \in \mathcal{D}_s\}$ is the number of times that item a is displayed by π_0 during the session.

In contrast to the counterfactual estimator presented in equation (2), the unbiased estimator derived from the sequential PBM has the ability to apply counterfactual adjustments to a specific item a that receives a positive reward within a session. However, an extra normalization step is necessary since a may appear in the displayed list for multiple rounds within a session. The proof of unbiasedness is presented as follows.

Proof. The value of policy π is the expectation of its total reward earned during the session. Thus

$$V_{\text{session}}(\pi) = \mathbb{E}_{\tilde{\mathbf{A}}}\left[\sum_{s=1}^{H}\sum_{k=1}^{n_s} \lambda_{s,k}p_{s,k}\theta_{A_{s,k}}\right]. \tag{5}$$

The second equality is from the definition of $\tilde{Y}_{s,k}$ and Assumption (3). From the structure of the position-based model, for any attraction probabilities θ, lists A_s in round s and $\tilde{\mathbf{A}}_k$ in round z, and positions k and ℓ, we have

$$p_{s,k}\theta_{A_{s,k}} = p_{s,k}\sum_{z=1}^{H}\sum_{\ell=1}^{n_z} \frac{1}{p_{z,\ell}} \frac{\mathbb{1}\{A_{s,k} = \tilde{A}_{z,\ell}\}}{C(A_{s,k})} p_{z,\ell}\theta_{\tilde{A}_{z,\ell}},$$

where we divide by $C(A_{s,k})$ to account for the fact that item $A_{s,k}$ could have been displayed in multiple rounds, and replace $\theta_{A_{s,k}}$ by $\theta_{\tilde{A}_{z,\ell}}$. Because this identity holds for any $\tilde{\mathbf{A}}_z$, it also holds in expectation over $\tilde{\mathbf{A}}_z$. Since $\tilde{\mathbf{A}}_z \sim \pi_0(\cdot \mid \mathbf{X}_z)$, and the sequence of contexts does not depend on earlier recommended lists, the counterfactual adjustment can be made jointly over all rounds and positions. Substituting it into (5), we obtain

$$\begin{aligned} V_{\text{session}}(\pi) &= \mathbb{E}_{\mathbf{A},\tilde{\mathbf{A}}}\left[\sum_{s,z=1}^{H}\sum_{k,\ell=1}^{n_s,n_z} \frac{\lambda_{s,k}p_{s,k}}{p_{z,\ell}} \frac{\mathbb{1}\{A_{s,k} = \tilde{A}_{z,\ell}\}}{C(A_{s,k})} p_{z,\ell}\theta_{\tilde{\mathbf{A}}_{z,\ell}}\right] \\ &= \mathbb{E}_{\mathbf{A},\tilde{\mathbf{A}}}\left[\sum_{s,z=1}^{H}\sum_{k,\ell=1}^{n_s,n_z} \frac{\lambda_{s,k}p_{s,k}}{p_{z,\ell}} \frac{\mathbb{1}\{A_{s,k} = \tilde{A}_{z,\ell}\}}{C(A_{s,k})} \tilde{Y}_{z,\ell}\right]. \end{aligned}$$

This concludes the proof. □

We also derive the gradient of the above session counterfactual DCG below.

Proposition 3.2. *The gradient of the session counterfactual DCG is*

$$\nabla V(\pi) = \mathbb{E}_{\mathbf{A},\tilde{\mathbf{A}}}\left[\sum_{s,z=1}^{H}\sum_{k,\ell=1}^{n_s,n_z} \frac{\lambda_{s,k}p_{s,k}}{p_{z,\ell}} \frac{\mathbb{1}\{A_{s,k} = \tilde{A}_{z,\ell}\}}{C(A_{s,k})} \tilde{Y}_{z,\ell}\nabla \log \pi(\mathbf{A}_{s,1:k} \mid \mathbf{X}_s)\right].$$

Proof. The gradient can be derived using the score identity, which states that $\nabla_\theta f(\theta) = f(\theta)\nabla_\theta \log f(\theta)$ holds for any non-negative function f and its parameters θ. Specifically, the gradient decomposes as

$$\begin{aligned} \nabla V(\pi) &= \nabla \mathbb{E}_{\tilde{\mathbf{A}}}\left[\sum_{s,z=1}^{H}\sum_{k,\ell=1}^{n_s,n_z}\sum_{\mathbf{A}_{s,1:k}} \pi(\mathbf{A}_{s,1:k}|\mathbf{X}_s) \cdot \frac{\lambda_{s,k}p_{s,k}}{p_{z,\ell}} \frac{\mathbb{1}\{A_{s,k} = \tilde{A}_{z,\ell}\}}{C(A_{s,k})} \tilde{Y}_{z,\ell}\right] \\ &= \mathbb{E}_{\tilde{\mathbf{A}}}\left[\sum_{s,z=1}^{H}\sum_{k,\ell=1}^{n_s,n_z}\sum_{\mathbf{A}_{s,1:k}} \nabla\pi(\mathbf{A}_{s,1:k}|\mathbf{X}_s) \cdot \frac{\lambda_{s,k}p_{s,k}}{p_{z,\ell}} \frac{\mathbb{1}\{A_{s,k} = \tilde{A}_{z,\ell}\}}{C(A_{s,k})} \tilde{Y}_{z,\ell}\right], \end{aligned}$$

where the first equality follows from the observation that $A_{s,k}$ only depends on the first k items in $\mathbf{A}_s$ and context $\mathbf{X}_s$. Then, for any list $\mathbf{A}_s$ and position k in it, we have

$$\nabla\pi(\mathbf{A}_{s,1:k} \mid \mathbf{X}_s) = \pi(\mathbf{A}_{s,1:k} \mid \mathbf{X}_s)\nabla \log \pi(\mathbf{A}_{s,1:k} \mid \mathbf{X}_s).$$

which is derived using the score identity. This concludes the proof. □

3.2 Sequential C-LTR vs. C-LTR

In this subsection, we discuss the advantages of *sequential* C-LTR over C-LTR and the complexity of the counterfactual evaluation.

Improving session-level ranking performance. The above estimator for sequential C-LTR in (4) would be a general sequential position-based IPS estimator of additive session-level ranking metrics when considering the cumulative reward of a search session as the objective, i.e., $\lambda_{s,k} \equiv 1$. The discount function $p_{s,k}$ is formulated based on our sequential user behavior model. It is worth noting that item a can be recommended multiple times within a session, necessitating normalization by $C(a)$.

Data augmentation using session information. A good way to interpret the unbiased estimator (4) is by propagating feedback across queries. For instance, if item a receives a positive reward in round z at position ℓ, the above estimator assigns pseudo rewards to item a in other queries, with counterfactual adjustments, if a

is present in their matching sets. As a result, session-level training utilizing the above estimator would obtain a larger number of positive labels compared to query-level training for LTR tasks, which would enhance the ranking performance. Furthermore, the increased complexity stemming from the sequential counterfactual evaluation in (4) is directly related to the frequency of items with positive rewards appearing in the display lists corresponding to the search session query chain. In other words, the proposed sequential C-LTR incorporates more search query samples into training while maintaining the same per-sample computational complexity as the conventional non-sequential C-LTR in (2).

4 EXPERIMENTS

In this section, we provide *preliminary* experimental findings based on synthetic data as well as real-world e-commerce data. Conducting extensive experiments on large-scale datasets poses certain challenges due to the *absence* of public e-commerce search session data containing search query chains, sequences of displayed ranked lists where the same item receives a reward in response to a later query, customer actions such as purchases, and expressive features. Hence, the experiments presented here serve as a catalyst for inspiring future research endeavors.

Policy Optimization. Our goal is to learn a ranking policy to optimize the value function using the offline data. The proposed counterfactual estimators of the value function in (2) and (4) can be seen as the loss function computed evaluated on the collected samples. To adopt gradient-based optimizers for training, we employ a parameterized ranking policy $\pi(\mathbf{A}|\mathbf{X}) = \pi(\mathbf{A}|\mathbf{X}, w)$ that is differentiable with respect to a feature vector w. Specifically, for random ranking policies, the softmax Plackett-Luce distribution [23, 24] can be considered, for which the conditional probability that item A_k is displayed at position k given a fixed list of higher ranked items $\mathbf{A}_{1:k-1}$ is given by

$$\pi(A_k \mid \mathbf{A}_{1:k-1}, \mathbf{X}; w) = \frac{\exp\{f(A_k, \mathbf{X}; w)\}}{\sum\limits_{a \in \mathcal{D}_\mathbf{X} \setminus \mathbf{A}_{1:k-1}} \exp\{f(a, \mathbf{X}; w)\}},$$

where $f(a, \mathbf{X}; w)$ is a learnable score for item a given context $\mathbf{X}$ parametrized by w. For non-differentiable deterministic ranking policies, the relaxed sorting operator, such as NeuralSort [10] and SoftSort [26], can be adopted to approximate the gradient. In the following experiments, we consider three LTR methods for deterministic ranking policies:

(i) PG: a policy gradient-based method to maximize $\hat{V}_{\text{query}}(\pi)$ in (2) with π being parametrized by NeuralSort, which serves a strong counterfactual baseline. This method is equivalent to a IPS-variant of PiRank [29] that optimizes the IPS-weighted IR metrics via differentiable sorting using NeuralSort.
(ii) Seq-PG: a policy gradient-based method to maximize $\hat{V}_{\text{session}}(\pi)$ in (4) with π being parametrized by NeuralSort. This approach can be seen as an extension of PiRank [29] aimed at optimizing counterfactual session-level IR metrics based on sPBM.
(iii) ListNet [4, 5]: a popular listwise approach with softmax cross entropy loss, which has consistently been shown to perform well for ranking problems and serves a strong baseline in supervised learning [27].

It is important to highlight that the three mentioned approaches primarily differ from each other solely in their loss functions. We refer to the first two approaches as PG and Seq-PG, as they involve utilizing policy gradient methods to optimize the derived counterfactual estimators of their value functions. The gradient is computed using automatic differentiation within the PyTorch framework.

Metrics. We evaluate the ranking performance using both conventional ranking metrics, such as DCG with traditional weights $1/\log_2(k+1)$, and counterfactual metrics $\hat{V}_{\text{query}}$ and $\hat{V}_{\text{session}}$ provided in (2) and (4) with $\lambda_k = \lambda_{s,k} = 1$, which we name as C-DCG and SC-DCG respectively. Counterfactual evaluation using offline data is shown to be more consistent with online performance than traditional metrics [17].

Synthetic Data. To demonstrate the benefits of SC-LTR, we test PG, Seq-PG, and ListNet on synthetic search session data generated from Baidu-ULTR [35], a comprehensive search dataset containing 1.2 billion web search queries and extensive user feedback. In particular, Baidu-ULTR provides subsequent reformulated query tokens issued under the same search goal for each query record, which enables us to create search sessions satisfying Assumption (4) by joining queries with the same search mission.

We generate 2,754 web search sessions for illustration by joining queries in a small subset of Baidu-ULTR. Each search session consists of 2 search queries that share the same clicked URL in their ranked lists but may place this URL at different positions. In particular, the clicked URL is presented at a relatively lower position under the first query compared to its rank in the second list. Synthetic sessions of this form mimic the sequential search behavior in which users tend to reformulate queries with unsatisfactory rankings.

For each search session that is in the form of a query chain, we mark all clicked URLs in the intersection of all response ranked lists as relevant and all other URLs as irrelevant. For each query-URL pair, a 768-dimensional feature vector is generated by a pre-trained language model provided in [35][1]. A whitening transformation is applied to all feature vectors to equalize feature scales and remove redundancy. We simulate training session data and test session data from a generative model in which the user randomly starts a search session from pre-generated 2,754 sessions and the probability of observing a click over URL a at round s and position k is given by $p_{s,k}$ if a is relevant. Multilayer perceptron (MLP) models are trained using the Adam optimizer with constant learning rate 0.01 to minimize objectives. Both train and test datasets include 5,000 simulated sessions with $p_{s,k} = (0.7)^{s-1} \times (0.8)^{k-1}$. This probability is chosen for illustration purposes, and implies that the likelihood of observing a click on a relevant URL is higher at the top of SERP when using the reformulated query compared to the bottom of the results page for the initial query.

In Figure 2, we plot the evaluated metrics on test data against the number of training epochs over 10 independent runs. The results indicate that Seq-PG has relatively superior performance in maximizing the counterfactual metrics C-DCG and SC-DCG, especially for session-level evaluation. On the other hand, ListNet tends to fit

[1] https://github.com/ChuXiaokai/baidu_ultr_dataset

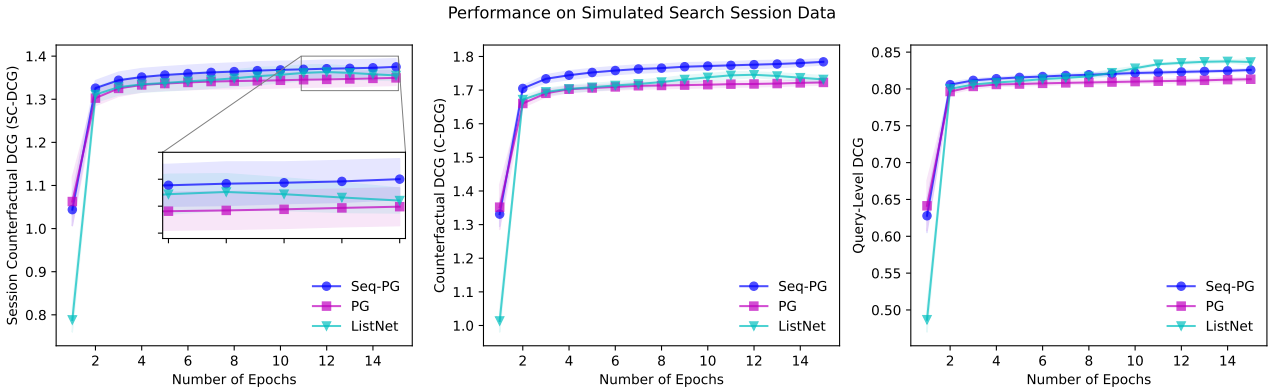

Figure 2: Comparisons of Seq-PG, PG, and ListNet on simulated session data. The shaded area indicates the confidence interval.

Table 1: Relative improvement of Seq-PG over ListNet on Proprietary Real-World Search Session Data.

	DCG	C-DCG	SC-DCG
PG	-3.98%	+0.15%	-1.14%
Seq-PG	-4.36%	+1.24%	+1.52%

the better ranking in terms of query-level non-counterfactual DCG, and its performance is relatively poorer in the other metrics.

Real-World Data. We also test the proposed SC-LTR method using proprietary real-world search session data collected from an e-commerce store within one week. The dataset is composed of 19,430 subsampled search sessions with lengths ranging from 2 to 6, and a total of 44,598 queries. In these sessions, the item purchased after the last reformulated query is also recommended in response to previous queries and is positioned lower on SERP. The query-item features are generated by a pre-trained transformer-based model. We trained MLP models to optimize three LTR objectives using Adam. The probabilities p_k and $p_{s,k}$ are selected as exponential decay functions, estimated from the marginal distribution of purchase labels. Table 1 presents the relative improvement of SC-LTR over ListNet, which also demonstrates the superiority of Seq-PG in finding a ranking policy improving session-level metrics.

5 RELATED WORK

Evaluation over search sessions were initiated through the Text REtrieval Conference (TREC) Session track [7]. The TREC Session tracks, which were conducted from 2011 to 2014, stand as the most commonly used web search session datasets. Subsequently, Chen et al. [8] presented the TianGong-ST dataset, a web search session dataset featuring both click-based and human-annotated relevance labels. However, these existing session datasets are not appropriate for our research due to its lack of expressive features and the absence of sequences of displayed ranked lists where the same item receives a reward in response to a subsequent query, which is a pattern commonly observed in e-commerce data.

It is noteworthy that the evaluation metrics utilized in the TREC Session track continue to rely on conventional evaluation measures originally developed for single-query searches. In this case, the primary task for these search engines is to deliver search results for the last reformulated query within each session in cases where no search results or clicks are provided. As a result, the evaluation process across the entire session is essentially equivalent, in terms of ranking, to evaluating solely the last query. As for efforts to evaluate ranking systems truly over query-sessions, Järvelin et al. [14] developed the evaluation measure session Discounted Cumulative Gain (sDCG), which is a generalization of the DCG evaluation measure [13]. Taking extending the cascade model [9] to query-sessions, Lipani et al. [19] derived a session-based Rank Biased Precision (RBP). Although our work shares a similar motivation for evaluating query-sessions, our primary focus in this study is on conducting counterfactual evaluations. Providing a comprehensive justification of the effectiveness of a session-level evaluation metric is not within the scope of our work.

6 CONCLUSION AND FUTURE WORK

We propose a novel problem of sequential counterfactual LTR which aims to improve the search experience in session-level ranking from offline data. We derive a counterfactual estimator for session-level ranking metrics under a sequential position-based model, and our preliminary experiments show that the proposed method outperforms existing methods in terms of counterfactual metrics. However, the assumption of sPBM can be violated in practice. For example, users' intentions can undergo changes throughout their interactions with the search engine. Providing assumption-free methods for SC-LTR is challenging, but it would be interesting to see future work to improve the session-level search experience from offline data. Overall, our work opens up a new direction for improving the session-level search experience and provides a useful framework for counterfactual evaluation in sequential settings.

REFERENCES

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

[2] Alexey Borisov, Ilya Markov, Maarten De Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*. 531–541.

[3] Alexey Borisov, Martijn Wardenaar, Ilya Markov, and Maarten De Rijke. 2018. A click sequence model for web search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 45–54.

[4] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval*. 75–78.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[6] Ben Carterette, Paul Clough, Mark Hall, Evangelos Kanoulas, and Mark Sanderson. 2016. Evaluating retrieval over sessions: The TREC session track 2011-2014. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 685–688.

[7] Ben Carterette, Evangelos Kanoulas, Mark M Hall, and Paul D Clough. 2014. Overview of the TREC 2014 Session Track.. In *TREC*.

[8] Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. TianGong-ST: A new dataset with large-scale refined real-world web search sessions. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2485–2488.

[9] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*. 87–94.

[10] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. 2018. Stochastic Optimization of Sorting Networks via Continuous Relaxations. In *International Conference on Learning Representations*.

[11] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. 55–64.

[12] Sharon Hirsch, Ido Guy, Alexander Nus, Arnon Dagan, and Oren Kurland. 2020. Query reformulation in E-commerce search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1319–1328.

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[14] Kalervo Järvelin, Susan L Price, Lois ML Delcambre, and Marianne Lykke Nielsen. 2008. Discounted cumulated gain based evaluation of multiple-query IR sessions. In *European Conference on Information Retrieval*. Springer, 4–15.

[15] 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*. 781–789.

[16] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2016. DCM bandits: Learning to rank with multiple clicks. In *International Conference on Machine Learning*. PMLR, 1215–1224.

[17] Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. 2015. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the 24th International Conference on World Wide Web*. 929–934.

[18] Shuai Li, Yasin Abbasi-Yadkori, Branislav Kveton, Shan Muthukrishnan, Vishwa Vinay, and Zheng Wen. 2018. Offline evaluation of ranking policies with click models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1685–1694.

[19] Aldo Lipani, Ben Carterette, and Emine Yilmaz. 2019. From a user model for query sessions to session rank biased precision (sRBP). In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 109–116.

[20] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. 2018. Breaking the curse of horizon: Infinite-horizon off-policy estimation. *Advances in Neural Information Processing Systems* 31 (2018).

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

[22] Haitao Mao, Lixin Zou, Yujia Zheng, Jiliang Tang, Xiaokai Chu, Jiashu Zhao, and Dawei Yin. 2022. Whole Page Unbiased Learning to Rank. *arXiv preprint arXiv:2210.10718* (2022).

[23] Harrie Oosterhuis. 2021. Computationally Efficient Optimization of Plackett-Luce Ranking Models for Relevance and Fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. ACM.

[24] Harrie Oosterhuis. 2022. Learning-to-Rank at the Speed of Sampling: Plackett-Luce Gradient Estimation With Minimal Computational Complexity. *arXiv preprint arXiv:2204.10872* (2022).

[25] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable unbiased online learning to rank. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 1293–1302.

[26] Sebastian Prillo and Julian Eisenschlos. 2020. Softsort: A continuous relaxation for the argsort operator. In *International Conference on Machine Learning*. PMLR, 7793–7802.

[27] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2021. Are neural rankers still outperformed by gradient boosted decision trees? (2021).

[28] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. 521–530.

[29] Robin Swezey, Aditya Grover, Bruno Charron, and Stefano Ermon. 2021. Pirank: Scalable learning to rank via differentiable sorting. *Advances in Neural Information Processing Systems* 34 (2021), 21644–21654.

[30] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 115–124.

[31] Zhenlei Wang, Shiqi Shen, Zhipeng Wang, Bo Chen, Xu Chen, and Ji-Rong Wen. 2022. Unbiased sequential recommendation with latent confounders. In *Proceedings of the ACM Web Conference 2022*. 2195–2204.

[32] Zhenlei Wang, Jingsen Zhang, Hongteng Xu, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Counterfactual data-augmented sequential recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 347–356.

[33] Chenyan Xiong and Jamie Callan. 2015. Esdrank: Connecting query and documents through external semi-structured data. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 951–960.

[34] Masrour Zoghi, Tomas Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2017. Online learning to rank in stochastic click models. In *International conference on machine learning*. PMLR, 4199–4208.

[35] Lixin Zou, Haitao Mao, Xiaokai Chu, Jiliang Tang, Wenwen Ye, Shuaiqiang Wang, and Dawei Yin. 2022. A Large Scale Search Dataset for Unbiased Learning to Rank. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.