Unbiased Learning to Rank: Counterfactual and Online Approaches

Harrie Oosterhuis* **Rolf Jagerman*** **Maarten de Rijke*,****

April 21, 2020

*University of Amsterdam

**Ahold Delhaize

oosterhuis@uva.nl, rolf.jagerman@uva.nl, derijke@uva.nl

The Web Conference 2020 Tutorial

Part 2: Counterfactual Learning to Rank

Part 2: Counterfactual Learning to Rank

This part will cover the following topics:

- **Counterfactual Evaluation**
  - Evaluating unbiasedly from historical interactions.
- **Propensity-weighted LTR**
  - Learning unbiasedly from historical interactions.
- **Estimating Position Bias**
- **Practical Considerations**
- **Related Work: Click Models**

Counterfactual Evaluation

Counterfactual Evaluation: Introduction

Evaluation is incredibly **important before deploying** a ranking system.

However, with the **limitations of annotated datasets**,
can we **evaluate** a ranker **without deploying** it or **annotated data**?

Counterfactual Evaluation:
Evaluate a new ranking function f_θ using **historical interaction data** (e.g., clicks) collected from a previously deployed ranking function f_{deploy}.

Counterfactual Evaluation: Full Information

If we **know** the **true relevance labels** ($y(d_i)$ for all i), we can compute any additive linearly decomposable IR metric as:

$$\Delta(f_\theta, D, y) = \sum_{d_i \in D} \lambda(\mathit{rank}(d_i \mid f_\theta, D)) \cdot y(d_i),$$

where λ is a rank weighting function, e.g.,

$$\begin{aligned}
\text{Average Relevant Position} \quad & \mathit{ARP} : \lambda(r) = r, \\
\text{Discounted Cumulative Gain} \quad & \mathit{DCG} : \lambda(r) = \frac{1}{\log_2(1 + r)}, \\
\text{Precision at } k \quad & \mathit{Prec@k} : \lambda(r) = \frac{\mathbf{1}[r \leq k]}{k}.
\end{aligned}$$

Counterfactual Evaluation: Full Information

$y(d_1) = 1$	Document d_1
$y(d_2) = 0$	Document d_2
$y(d_3) = 0$	Document d_3
$y(d_4) = 1$	Document d_4
$y(d_5) = 0$	Document d_5

Counterfactual Evaluation: Partial Information

We often do not know the true relevance labels $y(d_i)$, but can only observe implicit feedback in the form of, e.g., clicks:

- A click c_i on document d_i is a **biased and noisy indicator** that d_i is relevant
- A missing click does **not** necessarily indicate non-relevance.

Counterfactual Evaluation: Clicks

$y(d_1) = 1$	Document d_1	$c_1 = 1$
$y(d_2) = 0$	Document d_2	$c_2 = 0$
$y(d_3) = 0$	Document d_3	$c_3 = 1$
$y(d_4) = 1$	Document d_4	$c_4 = 0$
$y(d_5) = 0$	Document d_5	$c_5 = 0$

Counterfactual Evaluation: Clicks

Remember that there are many reasons why a click on a document may **not** occur:

- **Relevance**: the document may not be relevant.
- **Observance**: the user may not have examined the document.
- **Miscellaneous**: various random reasons why a user may not click.

Some of these reasons are considered to be:

- **Noise**: averaging over many clicks will remove their effect.
- **Bias**: averaging will **not** remove their effect.

Counterfactual Evaluation: Examination User Model

If we **only** consider **examination** and **relevance**, a user click can be modelled by:

- The probability of document d_i **being examined** $(o_i = 1)$ in a ranking R:

$$P(o_i = 1 \mid R, d_i).$$

- The probability of a **click** $c_i = 1$ on d_i given its **relevance** $y(d_i))$ and whether it was **examined** o_i:

$$P(c_i = 1 \mid o_i, y(d_i)).$$

- **Clicks only occur on examined documents**, thus the probability of a click in ranking R is:

$$P(c_i = 1 \wedge o_i = 1 \mid y(d_i), R) = P(c_i = 1 \mid o_i = 1, y(d_i)) \cdot P(o_i = 1 \mid R, d_i).$$

Counterfactual Evaluation: Naive Estimator

A **naive way** to estimate is to assume clicks are a unbiased relevance signal:

$$\Delta_{\mathit{NAIVE}}(f_\theta, D, c) = \sum_{d_i \in D} \lambda(\mathit{rank}(d_i \mid f_\theta, D)) \cdot c_i.$$

Even if **no click noise** is present: $P(c_i = 1 \mid o_i = 1, y(d_i)) = y(d_i)$, this estimator is **biased** by the examination probabilities:

$$\begin{aligned}\mathbb{E}_o[\Delta_{\mathit{NAIVE}}(f_\theta, D, c)] &= \mathbb{E}_o\left[\sum_{d_i : o_i = 1 \wedge y(d_i) = 1} \lambda(\mathit{rank}(d_i \mid f_\theta, D))\right] \\ &= \sum_{d_i : y(d_i) = 1} P(o_i = 1 \mid R, d_i) \cdot \lambda(\mathit{rank}(d_i \mid f_\theta, D)).\end{aligned}$$

Counterfactual Evaluation: Naive Estimator Bias

The biased estimator **weights documents** according to their **examination probabilities** in the ranking R displayed during **logging**:

$$\mathbb{E}_o[\Delta_{\mathit{NAIVE}}(f_\theta, D, c)] = \sum_{d_i : y(d_i)=1} P(o_i = 1 \mid R, d_i) \cdot \lambda(\mathit{rank}(d_i \mid f_\theta, D)).$$

In rankings, **documents at higher ranks** are more likely to be examined: **position bias**.

Position bias causes **logging-policy-confirming** behavior:

- Documents displayed at **higher ranks during logging** are incorrectly considered as **more relevant**.

Inverse Propensity Scoring

Counterfactual Evaluation: Inverse Propensity Scoring

Counterfactual evaluation accounts for bias using **Inverse Propensity Scoring (IPS)**:

$$\Delta_{IPS}(f_\theta, D, c) = \sum_{d_i \in D} \frac{\lambda(\mathit{rank}(d_i \mid f_\theta, D))}{P(o_i = 1 \mid R, d_i)} \cdot c_i,$$

where

- $\lambda(\mathit{rank}(d_i \mid f_\theta, D))$: (weighted) rank of document d_i by ranker f_θ,
- c_i: observed click on the document in the log,
- $P(o_i = 1 \mid R, d_i)$: examination probability of d_i in ranking R displayed during logging.

This is an **unbiased estimate** of any additive linearly decomposable IR metric.

Counterfactual Evaluation: Proof of Unbiasedness

If no click noise is present, this provides an **unbiased estimate**:

$$
\begin{aligned}
\mathbb{E}_o[\Delta_{\mathit{IPS}}(f_\theta, D, c)] &= \mathbb{E}_o\left[\sum_{d_i \in D} \frac{\lambda(\mathit{rank}(d_i \mid f_\theta, D))}{P(o_i = 1 \mid R, d_i)} \cdot c_i\right] \\
&= \mathbb{E}_o\left[\sum_{d_i : o_i = 1 \wedge y(d_i) = 1} \frac{\lambda(\mathit{rank}(d_i \mid f_\theta, D))}{P(o_i = 1 \mid R, d_i)}\right] \\
&= \sum_{d_i : y(d_i) = 1} \frac{P(o_i = 1 \mid R, d_i) \cdot \lambda(\mathit{rank}(d_i \mid f_\theta, D))}{P(o_i = 1 \mid R, d_i)} \\
&= \sum_{d_i \in D} \lambda(\mathit{rank}(d_i \mid f_\theta, D)) \cdot y(d_i) \\
&= \Delta(f_\theta, D, y).
\end{aligned}
$$

Counterfactual Evaluation: Robustness of Noise

So far we have **no click noise**: $P(c_i = 1 \mid o_i = 1, y(d_i)) = y(d_i)$.

However, the IPS approach still works without these assumptions, as long as:

$$y(d_i) > y(d_j) \Leftrightarrow P(c_i = 1 \mid o_i = 1, y(d_i)) > P(c_j = 1 \mid o_j = 1, y(d_j)).$$

Since we can prove **relative differences** are inferred unbiasedly:

$$\mathbb{E}_{o,c}[\Delta_{\mathit{IPS}}(f_\theta, D, c)] > \mathbb{E}_{o,c}[\Delta_{\mathit{IPS}}(f_{\theta'}, D, c)] \Leftrightarrow \Delta(f_\theta, D) > \Delta(f_{\theta'}, D).$$

Propensity-weighted Learning to Rank

Propensity-weighted Learning to Rank (LTR)

The inverse-propensity-scored estimator can unbiasedly estimate performance:

$$\Delta_{\mathit{IPS}}(f_\theta, D, c) = \sum_{d_i \in D} \frac{\lambda(\mathit{rank}(d_i \mid f_\theta, D))}{P(o_i = 1 \mid R, d_i)} \cdot c_i.$$

How do we **optimize** for this **unbiased performance estimate**?

- It is **not differentiable**.
- **Common problem for all ranking metrics**.

Upper Bound on Rank

Rank-SVM (Joachims, 2002) optimizes the following **differentiable upper bound**:

$$\mathit{rank}(d \mid f_\theta, D) = \sum_{d' \in R} \mathbb{1}[f_\theta(d) \leq f_\theta(d')]$$
$$\leq \sum_{d' \in R} \max(1 - (f_\theta(d) - f_\theta(d')), 0) = \overline{\mathit{rank}}(d \mid f_\theta, D).$$

Alternative choices are possible, i.e., a **sigmoid-like bound** (with parameter σ):

$$\mathit{rank}(d \mid f_\theta, D) \leq \sum_{d' \in R} \log_2(1 + \exp^{-\sigma(f_\theta(d) - f_\theta(d'))}).$$

Commonly used for pairwise learning, LambdaMart (Burges, 2010), and Lambdaloss (Wang et al., 2018c).

Propensity-weighted LTR: Average Relevance Position

Then for the Average Relevance Position metric:

$$\Delta_{\mathit{ARP}}(f_\theta, D, y) = \sum_{d_i \in D} \mathit{rank}(d_i \mid f_\theta, D) \cdot y(d_i).$$

This gives us an **unbiased estimator** and **upper bound**:

$$\begin{aligned}\Delta_{\mathit{ARP\text{-}IPS}}(f_\theta, D, c) &= \sum_{d_i \in D} \frac{\mathit{rank}(d_i \mid f_\theta, D)}{P(o_i = 1 \mid R, d_i)} \cdot c_i \\ &\leq \sum_{d_i \in D} \frac{\overline{\mathit{rank}}(d_i \mid f_\theta, D)}{P(o_i = 1 \mid R, d_i)} \cdot c_i,\end{aligned}$$

This upper bound is **differentiable** and **optimizable** by stochastic gradient descent or Quadratic Programming, i.e., Rank-SVM (Joachims, 2006).

Propensity-weighted LTR: Additive Metrics

A similar approach can be applied to **additive metrics** (Agarwal et al., 2019a).

If λ is a **monotonically decreasing** function:

$$x \leq y \Rightarrow \lambda(x) \geq \lambda(y),$$

then:

$$\mathit{rank}(d \mid \cdot) \leq \overline{\mathit{rank}}(d \mid \cdot) \Rightarrow \lambda(\mathit{rank}(d \mid \cdot)) \geq \lambda(\overline{\mathit{rank}}(d \mid \cdot)).$$

This provides a **lower bound**, for instance for Discounted Cumulative Gain (DCG):

$$\frac{1}{\log_2(1 + \mathit{rank}(d \mid \cdot))} \geq \frac{1}{\log_2(1 + \overline{\mathit{rank}}(d \mid \cdot))}.$$

Propensity-weighted LTR: Discounted Cumulative Gain

Then for the Discounted Cumulative Gain metric:

$$\Delta_{\mathit{DCG}}(f_\theta, D, y) = \sum_{d_i \in D} \log_2(1 + \mathit{rank}(d_i \mid f_\theta, D))^{-1} \cdot y(d_i).$$

This gives us an **unbiased estimator** and **lower bound**:

$$\begin{aligned}\Delta_{\mathit{DCG\text{-}IPS}}(f_\theta, D, c) &= \sum_{d_i \in D} \frac{\log_2(1 + \mathit{rank}(d_i \mid f_\theta, D)^{-1}}{P(o_i = 1 \mid R, d_i)} \cdot c_i \\ &\geq \sum_{d_i \in D} \frac{\log_2(1 + \overline{\mathit{rank}}(d_i \mid f_\theta, D)^{-1}}{P(o_i = 1 \mid R, d_i)} \cdot c_i.\end{aligned}$$

This lower bound is **differentiable** and **optimizable** by stochastic gradient descent or the Convex-Concave Procedure (Agarwal et al., 2019a).

Propensity-weighted LTR: Walkthrough

Overview of the approach:

- Obtain a **model of position bias**.
- Acquire a **large click-log**.
- Then for every click in the log:
  - Compute the **propensity of the click**:

$$P(o_i = 1 \mid R, d_i).$$

  - Calculate the **gradient** of the **bound** on the **unbiased estimator**:

$$\nabla_\theta \left[\frac{\overline{\mathit{rank}}(d_i \mid f_\theta, D)}{P(o_i = 1 \mid R, d_i)} \right].$$

  - **Update the model** f_θ by adding/subtracting the gradient.

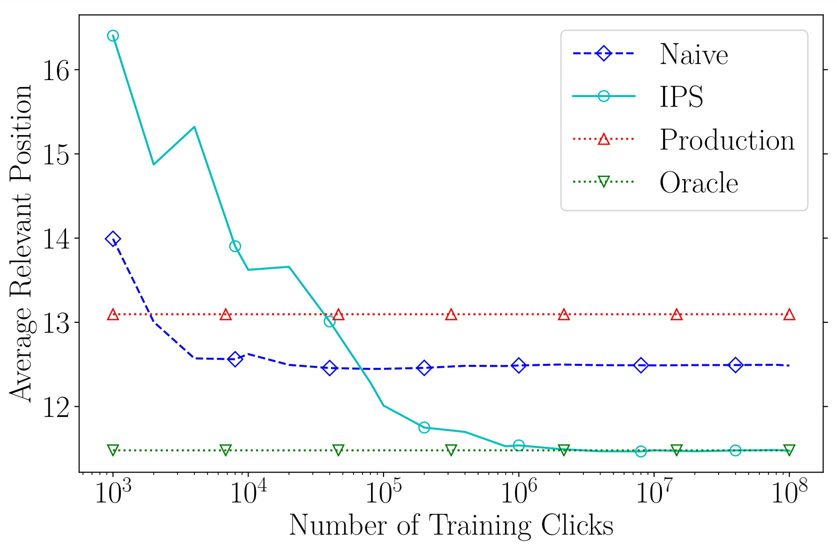
Propensity-weighted LTR: Semi-synthetic Experiments

Unbiased LTR methods are commonly **evaluated** through **semi-synthetic experiments** (Joachims, 2002; Agarwal et al., 2019a; Jagerman et al., 2019).

The experimental setup:

- Traditional LTR dataset, e.g., Yahoo! Webscope (Chapelle and Chang, 2011).
- Simulate queries by uniform sampling from the dataset.
- Create a ranking according to a baseline ranker.
- Simulate clicks by modelling:
  - **Click Noise**, e.g., 10% chance of clicking on a non-relevant document.
  - **Position Bias**, e.g., $P(o_i = 1 \mid R, d_i) = \frac{1}{\textit{rank}(d|R)}$.
- Hyper-parameter tuning by unbiased evaluation methods.

Propensity-weighted LTR: Results



Estimating Position Bias

Estimating Position Bias

So far we have seen how to:

- Perform **Counterfactual Evaluation** with **unbiased estimators**.
- Perform **Counterfactual LTR** by optimizing **unbiased estimators**.

At the core of these methods is the propensity score: $P(o_i = 1 \mid R, d_i)$, which helps to remove bias from user interactions.

In this section, we will show how this **propensity score** can be **estimated** for a specific kind of bias: **position bias**.

Estimating Position Bias

Recall that position bias is a form of bias where higher positioned results are more likely to be observed and therefore clicked.

Assumption: The **observation probability** only depends on the rank of a document:

$$P(o_i = 1 \mid i).$$

The objective is now to **estimate**, for each rank i, the propensity $P(o_i = 1 \mid i)$.

This user model was first formalized by Craswell et al. (2008).

Estimating Position Bias

RandTop-n Algorithm:

Document d_1	Document d_3	Document d_2	Rank 1
Document d_2	Document d_4	Document d_1	Rank 2
Document d_3	Document d_1	Document d_4	Rank 3
Document d_4	Document d_2	Document d_3	Rank 4

Estimating Position Bias

RandTop-n Algorithm:

1. Repeat:
   - Randomly shuffle the top n items
   - Record clicks
2. Aggregate clicks per rank
3. Normalize to obtain propensities $p_i \propto P(o_i \mid i)$

Note: we only need propensities proportional to the true observation probability for learning.

Estimating Position Bias

Uniformly **randomizing** the top n results may negatively impacts users during data logging.

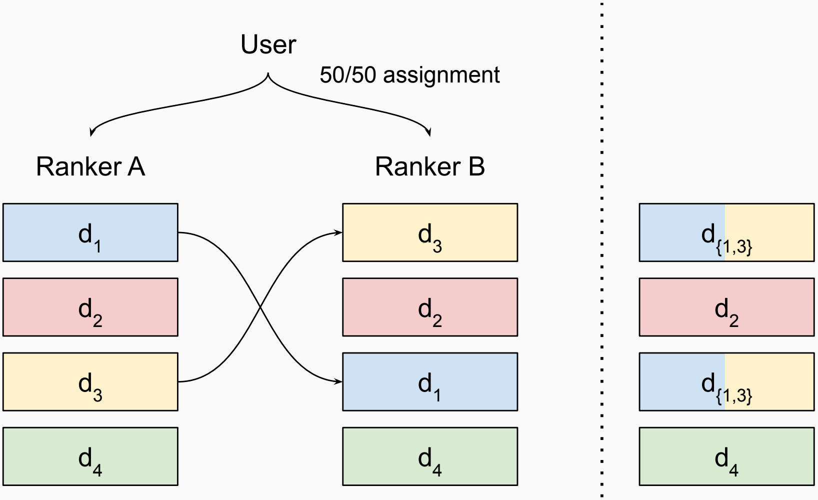
There are various methods that minimize the impact to the user:

- **RandPair:** Choose a pivot rank k and only swap a random other document with the document at this pivot rank (Joachims et al., 2017b).
- **Interventional Sets:** Exploit inherent "randomness" in data coming from multiple rankers (e.g., A/B tests in production logs) (Agarwal et al., 2017).

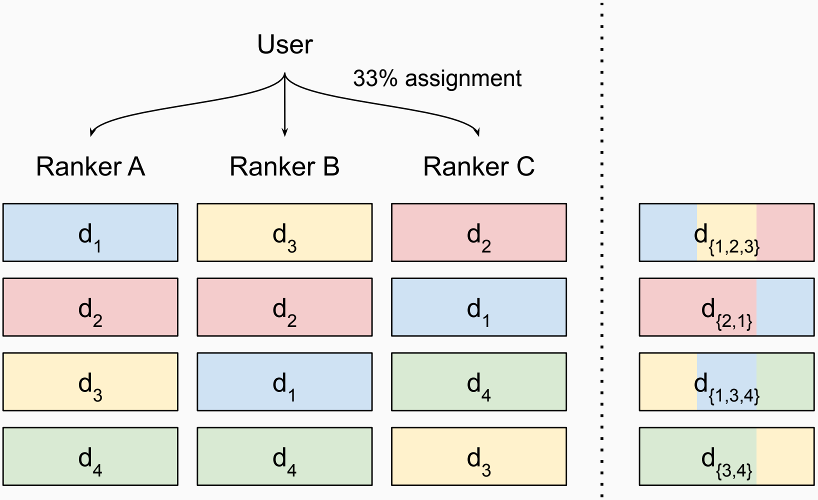
Intervention Harvesting

- As we have seen, to measure position bias, the most straightforward approach is to perform randomization.
- Naturally, we want to avoid randomizing because this negatively affects the end-user experience.
- **Main idea**: In real-world production systems many (randomized) interventions take place due to *A/B tests*. Can we use these interventions instead?
- This approach is called *intervention harvesting* (Agarwal et al. (2017); Fang et al. (2019); Agarwal et al. (2019c))

Intervention Harvesting



Intervention Harvesting



Jointly Learning and Estimating

Jointly Learning and Estimating

In the previous sections we have seen:

- Counterfactual ranker evaluation with unbiased estimators.
- Counterfactual LTR by optimizing unbiased estimators.
- Estimating propensity scores through randomization.

Instead of treating **propensity estimation** and **unbiased learning to rank** as two separate tasks, recent work has explored **jointly learning rankings and estimating propensities**.

Jointly Learning and Estimating

Recall that the probability of a click can be decomposed as:

$$\underbrace{P(c_i = 1 \wedge o_i = 1 \mid y(d_i), R)}_{\text{click probability}} = \underbrace{P(c_i = 1 \mid o_i = 1, y(d_i))}_{\text{relevance probability}} \cdot \underbrace{P(o_i \mid R, d_i)}_{\text{observation probability}}.$$

In the previous sections we have seen that, if the **observation probability** is known, we can find an unbiased estimate of relevance via IPS.

Jointly Learning and Estimating

It is possible to **jointly learn and estimate** by iterating two steps:

1. Learn an optimal ranker given a correct propensity model:

$$\underbrace{P(c_i = 1 \mid o_i = 1, y(d_i))}_{\text{relevance probability}} = \frac{P(c_i = 1 \wedge o_i = 1 \mid y(d_i), R)}{P(o_i \mid R, d_i)}.$$

2. Learn an optimal propensity model given a correct ranker:

$$\underbrace{P(o_i \mid R, d_i)}_{\text{observation probability}} = \frac{P(c_i = 1 \wedge o_i = 1 \mid y(d_i), R)}{P(c_i = 1 \mid o_i = 1, y(d_i))}.$$

Jointly Learning and Estimating

- Given an accurate **model of relevance**, it is possible to find an accurate **propensity model**, and vice versa.
- This approach requires **no randomization**.
- Recent work has solved this via either an **Expectation-Maximization approach** (Wang et al. (2018b)) or a **Dual Learning Objective** (Ai et al. (2018)).

Addressing Trust Bias

Addressing Trust Bias

In recent work Agarwal et al. (2019b) also address trust bias.

Trust bias:

- Users more often **overestimate** the **relevance** of **higher** ranked documents, and more often **underestimate** the **relevance** of **lower** ranked documents (Agarwal et al., 2019b; Joachims et al., 2017a).

Trust bias is related to position bias but involves more than just examination bias.

Modelling Trust Bias

Clicks are now modelled on the **perceived relevance** $\tilde{y}(d_i)$ instead of the **actual relevance** $y(d_i)$:

$$P(c_i \mid d_i, R, y) = P(\tilde{y}(d_i) = 1 \mid y(d_i), R) \cdot P(o_i = 1 \mid R, d_i).$$

Agarwal et al. (2019b) model the perceived relevance conditioned on the actual relevance and **display position** *rank*$(d_i, R) = k$:

$$P(\tilde{y}(d_i) = 1 \mid y(d_i) = 1, k) = \epsilon_k^+,$$
$$P(\tilde{y}(d_i) = 1 \mid y(d_i) = 0, k) = \epsilon_k^-.$$

Correcting for Trust Bias

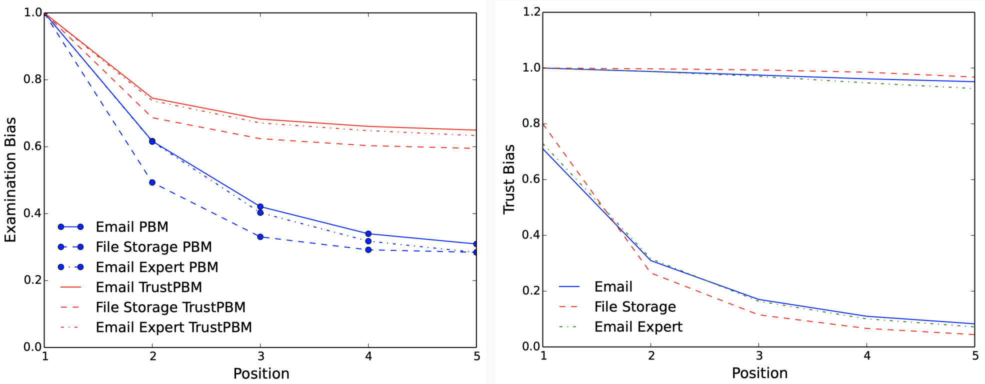
The new estimator becomes:

$$\Delta_{\textit{Bayes-IPS}}(f_\theta, D, c) = \sum_{d_i \in D} P(y(d_i) = 1 | c_i = 1, k) \cdot \frac{\lambda(\textit{rank}(d_i \mid f_\theta, D))}{P(o_i = 1 \mid R, d_i)} \cdot c_i$$

$$= \sum_{d_i \in D} \frac{\epsilon_k^+}{\epsilon_k^+ + \epsilon_k^-} \cdot \frac{\lambda(\textit{rank}(d_i \mid f_\theta, D))}{P(o_i = 1 \mid R, d_i)} \cdot c_i.$$

The ϵ values can **not be inferred** through **randomization experiments**, but can be estimated through **EM-optimization**.

Disentangled Examination and Trust Bias



If trust bias is **not modeled separately**, then the estimated examination bias will be affected by it. This may explain why the **performance gains** are **somewhat limited**.

Image credits: (Agarwal et al., 2019b).

Practical Considerations

Practical Considerations

Practitioners of counterfactual LTR systems will run into the problem of **high variance**.

High variance can be due to many factors:

- Not enough training data
- Extreme position bias and very small propensity
- Large amounts of noisy clicks on documents with small propensity

The usual suspect is one or a few data points with extremely small propensity that overpower the rest of the data set.

Practical Considerations

A typical solution to **high variance** is to apply **propensity clipping**.

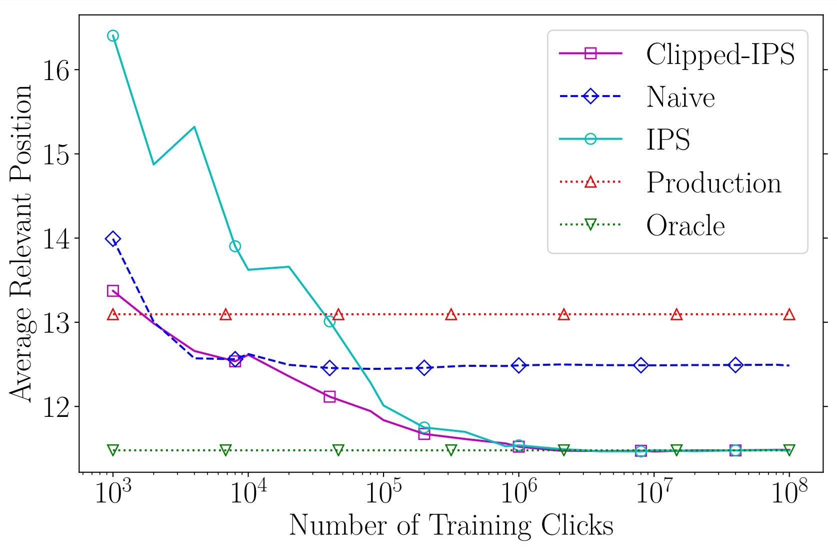
Propensity clipping: Bound the *propensity*, to prevent any single sample from overpowering the rest of the data set:

$$\Delta_{\textit{Clipped-IPS}}(f_\theta, D, c) = \sum_{d_i \in D} \frac{\lambda(\textit{rank}(d_i \mid f_\theta, D))}{\max\{\tau, P(o_i = 1 \mid R, d_i)\}} \cdot c_i.$$

This solution trades off bias for variance: it will introduce some amount of bias but can substantially reduce variance.

Note that when $\tau = 1$, we obtain the biased naive estimator.

Practical Considerations



Comparison to Supervised LTR

Comparison to Supervised LTR

Supervised LTR:

- Uses **manually annotated labels**:
  - expensive to create,
  - impossible in many settings,
  - often misaligned with actual user preferences.
- Optimization is widely studied and very effective w.r.t. evaluation on annotated labels.
- Often unavailable for practitioners.

Counterfactual LTR:

- Uses **click logs**:
  - available in abundant quantities,
  - effectively no cost,
  - contains **noise** and **biases**.
- **Noise**: amortized over large numbers of clicks.
- **Biases**:
  - position bias mitigated with inverse propensity scoring.
  - other biases are an active area of research.