Optimizing a Ranking System with User Interaction Logs: Counterfactual Learning to Rank

Harrie Oosterhuis

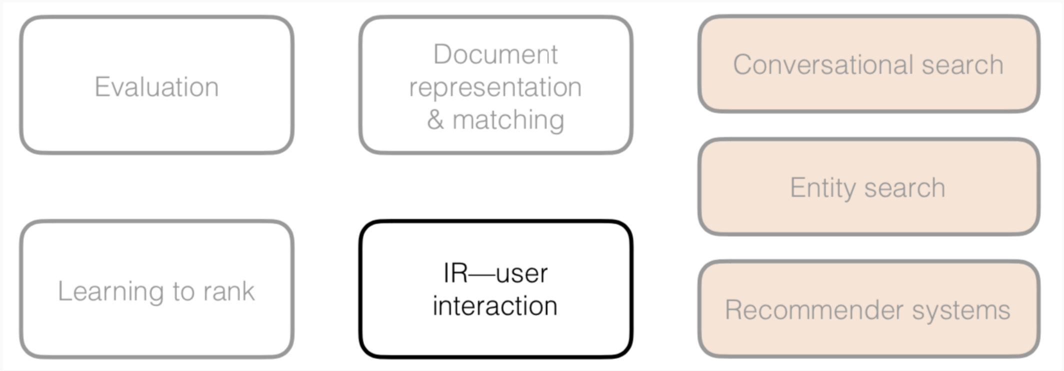
February 25, 2020

University of Amsterdam

oosterhuis@uva.nl

Based on the SIGIR 2019 Tutorial:
Unbiased Learning to Rank: Counterfactual and Online Approaches
Harrie Oosterhuis, Rolf Jagerman, Maarten de Rijke

Course Overview



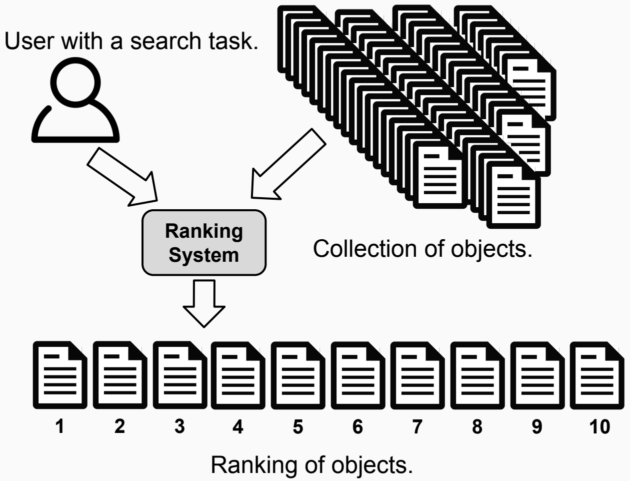
Introduction: Ranking Systems

Ranking Systems

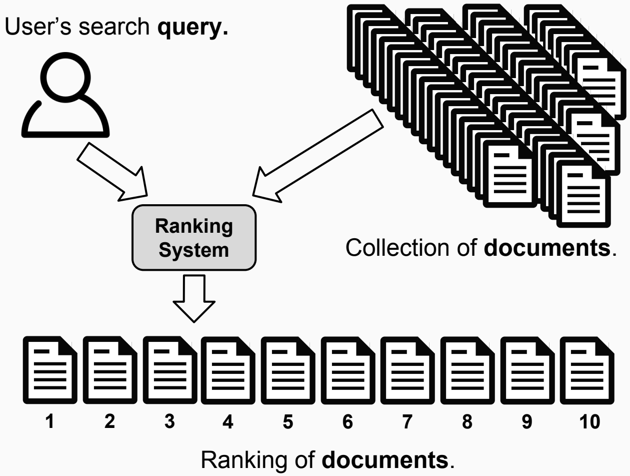
Let's go back to the beginning:

- Ranking systems are vital for **making large document collections accessible**.
- They can present users with **a small comprehensible selection** out of **millions of unordered results**.
- Search and recommendation are **practically everywhere**.

Ranking Systems: Schematic Example



Ranking Systems: Schematic Example Naming



Supervised Learning to Rank

Learning to Rank in Information Retrieval

Learning to Rank is a **core task** in informational retrieval:

- Key component for **search** and **recommendation**.

Different from **regression** where we want to scores to **match labels**,
for document d and relevance function $y(d)$ and a ranking function $f_\theta(d) \in \mathbb{R}$:

$$f_\theta(d) = y(d).$$

In **learning to rank**, we only care about the **ordering** according to f_θ:

$$y(d_1) > y(d_2) \rightarrow f_\theta(d_1) > f_\theta(d_2).$$

Supervised Learning to Rank

Traditionally learning to rank is **supervised** through **annotated datasets**:

- **Relevance annotations** for query-document pairs provided by **human judges**.

Over the years several limitations of annotated datasets have become apparent,

can you think of some limitations?

Limitations of the Annotated Datasets

Some of the most substantial limitations of **annotated datasets** are:

- **expensive** to make (Qin and Liu, 2013; Chapelle and Chang, 2011).
- **unethical** to create in **privacy-sensitive settings** (Wang et al., 2016).
- **impossible** for small scale problems, e.g., **personalization**.
- **stationary**, cannot capture **future changes in relevancy** (Lefortier et al., 2014).
- **not necessarily aligned with actual user preferences** (Sanderson, 2010),

  i.e., annotators and users often disagree.

Limitations of the Supervised Approach

Annotated datasets are **valuable** and have an **important place in research and development**.

However, the supervised approach is:

- **Unavailable** for practitioners without a **considerable budget**.
- **Impossible** for certain ranking problems.
- Often **misaligned** with *true* user preferences.

Therefore, there is a **need** for an **alternative** learning to rank approach.

solution:用user interaction代替人工 annotation

Learning from User Interactions

Learning from User Interactions: Advantages

Learning from user interactions solves the problems of annotations:

- Interactions are **virtually free** if you have users.
- User **behavior** is indicative of their **preferences**.

User interactions also bring their **own difficulties**:

- Interactions give **implicit feedback**.

Learning from User Interactions: Difficulties

没有 relevancy，user也会click
有relevancy，user却不click
noise are not bad. we can remove noise by taking avg of all data points to get a mean

but we can not solve bias by taking the mean

User interactions bring their **own difficulties**:

- **Noise:**
  - Users click for **unexpected reasons**.
  - Often clicks occur **not because** of relevancy.
  - Often clicks do not occur **despite** of relevancy.
- **Bias:** Interactions are affected by **factors other than relevancy**:
  - **Position**
  - **Item sel**
  - **Presenta**
    different
  - . . .

bias的定义和来源
position bias: these docs are clicked not because they are rel, but because they are positioned on top of the webpage

item selection bias: users dont see all the rel doc. some doc are rel, but dont be shown to the users. so these rel doc dont have user interations (clicks)

The Golden Triangle

position bias

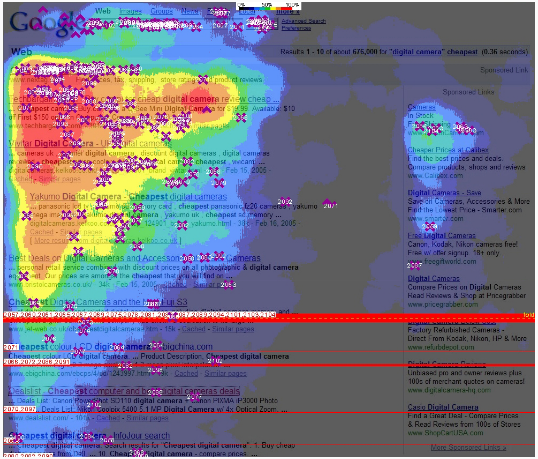


Image source: http://www.mediative.com/

Learning from User Interactions: Goal

there are always bias in clicks, we don't want to learn bias. we only want to learn how users interact with rel doc

note: unbiased object does not exist in reality. but exist in math

Goal of unbiased learning to rank:

- Optimize a ranker w.r.t. **relevance preferences** of users from their interactions.
- **Avoid** being **biased by other factors** that influence interactions.

This Lecture

There are currently **two main approaches** to Unbiased Learning to Rank:

Online Learning to Rank

- Learning by **directly interacting with users.**
- Handle biases through **randomization of displayed results**.

Counterfactual Learning to Rank

- Learning from **historical interactions.**
- Use a **model of user behavior** to correct for biases.

We will discuss the latter.

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}.

known:
current ranking func (has ployed) & user interaction on the current ranking func
new ranking func (not deployed)
want to know: how effective the new ranking func is

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: true relevant func

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

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

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

Counterfactual Evaluation: Full Information

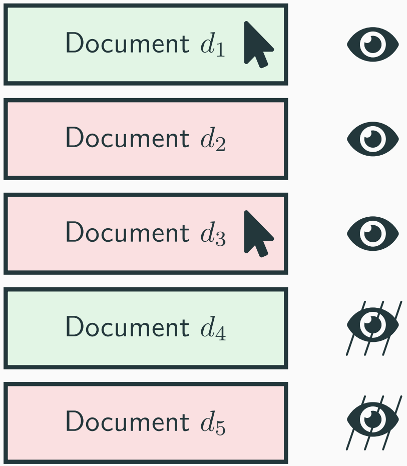
$y(d_1) = 1$	🟩 Document d_1
$y(d_2) = 0$	🟥 Document d_2
$y(d_3) = 0$	🟥 Document d_3
$y(d_4) = 1$	🟩 Document d_4
$y(d_5) = 0$	🟥 Document d_5

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



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

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:

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

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:

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

Counterfactual Evaluation: Naive Estimator Bias

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

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

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

What effect does this have on the evaluation?

Counterfactual Evaluation: Naive Estimator Bias

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

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

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**.

1h06min
make sure each doc is examined with equal prob. boost the prob to exam bottom ranked doc. reduce the prob to exam top ranked doc

only one goal to use IPS: fix the bias, if bias not existing, we can just use clicks as labels.

Inverse Propensity Scoring

Counterfactual Evaluation: Inverse Propensity Scoring

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

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

- $\lambda(\mathit{rank}(d_i \mid f_\theta, D))$: (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.

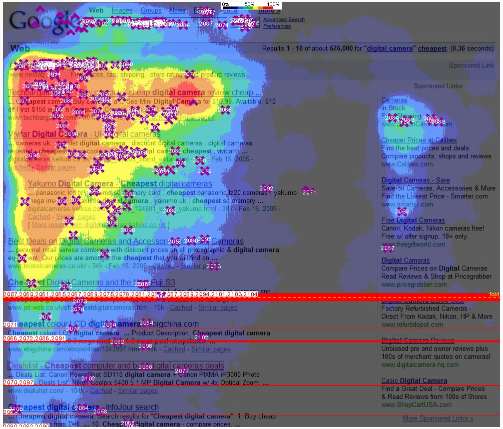
This is an **unbiased estimate** of any additive linearly decomposable IR metric.

Counterfactual Evaluation: Proof of Unbiasedness

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

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

Remember the Golden Triangle?



The IPS estimator weights clicks inversely proportional to the examination probabilities.

Image source: `http://www.mediative.com/`

Counterfactual Evaluation: Robustness of Noise

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

However, the IPS approach still works without this assumption, 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:

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

1h17min

Propensity-weighted Learning to Rank

Propensity-weighted Learning to Rank (LTR)

metrics in IR are not differentiable. so we cant do Stochastic gradient descent

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

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

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

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

Upper Bound on Rank

for doc 1, how many doc is higher or equal than doc1? return 1 (only doc 1)

Rank-SVM (Joachims, 2002) optimize

use a hinge func to upper bound my ranking func.

1) my ranking func always < upper bound
2) the upper bound is differentiable
3) by minimizing upperbound, we minimize the true rank func

$$\mathit{rank}(d \mid f_\theta, D) = \sum_{d' \in R} \mathbb{1}[f_\theta(d$$

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

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., 2018b).

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**:

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

we minimize upper bound to minimize ARP

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

lambda is单调递减函数。 when its input is upper bound of ranking func, then lambda(ranking func) will get a lower bound, which is lambda (upper bound). we max true func, by maximizing lower bound. true func is not differentiable. but lower bound is differentiable.

为什么用单调减函数:in a ranked list, doc的排位数越低，我们想赋予这个doc的weight就越大

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

倾向加权

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**:
      $$\nabla_\theta \left[\frac{\lambda(\overline{\mathit{rank}}(d_i \mid f_\theta, D))}{P(o_i = 1 \mid R, d_i)} \right].$$
    - **Update the model** f_θ by adding/subtracting the gradient.

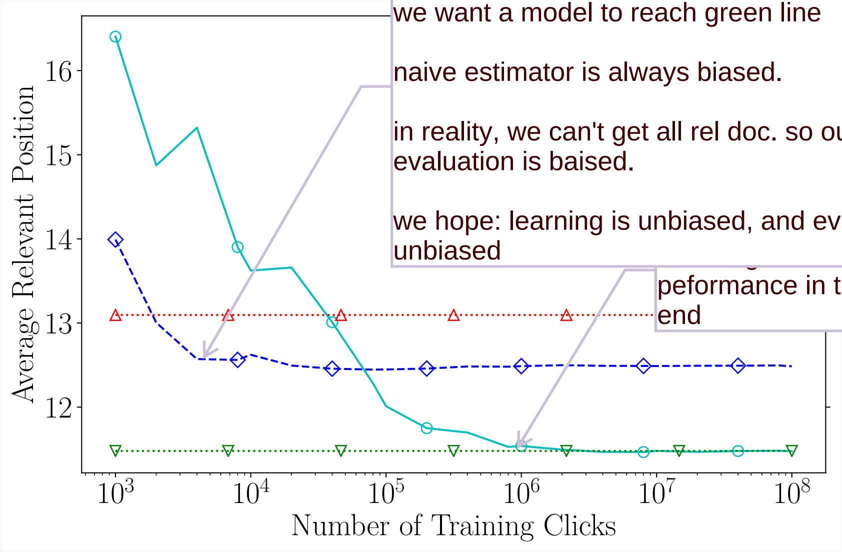
Propensity-weighted LTR: Semi-synthetic Experiments

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., $P(o_i = 1 \mid R, d_i) = \frac{1}{\mathit{rank}(d|R)}$.
- Hyper-parameter tuning by unbiased evaluation methods.

Propensity-weighted LTR: Results



for naive method, after it converges, no matter how many more clicks we provide to the model, the model's performance does not become better. we want a model to reach green line

naive estimator is always biased.

in reality, we can't get all rel doc. so our evaluation is baised.

we hope: learning is unbiased, and evaluation is unbiased

peformance in the end

So far so good

So far we have seen how to:

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

What essential part are we still missing?

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:

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

i: is the ranked position of this doc

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

propensity= prob of observation

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 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.

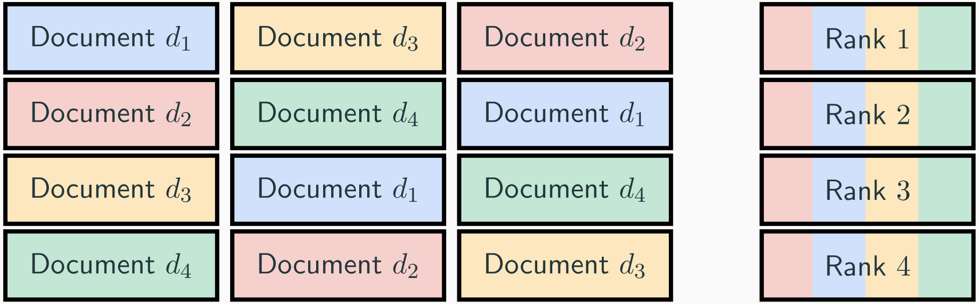
Assumption: The **observation probability** only depends on the rank of a document:

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

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

Estimating Position Bias

RandTop-n Algorithm:



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 Through Randomization

What is the downside of estimating propensities through randomization?

Estimating Position Bias

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

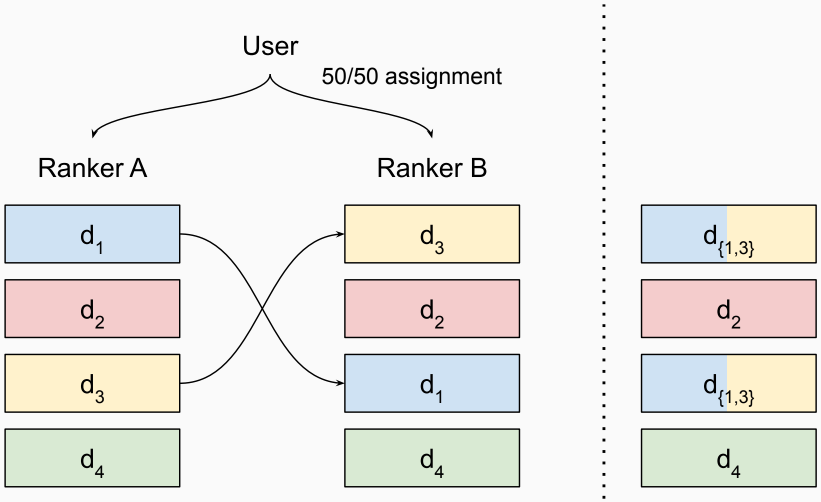
There are various methods that minimize the impact to the user:

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

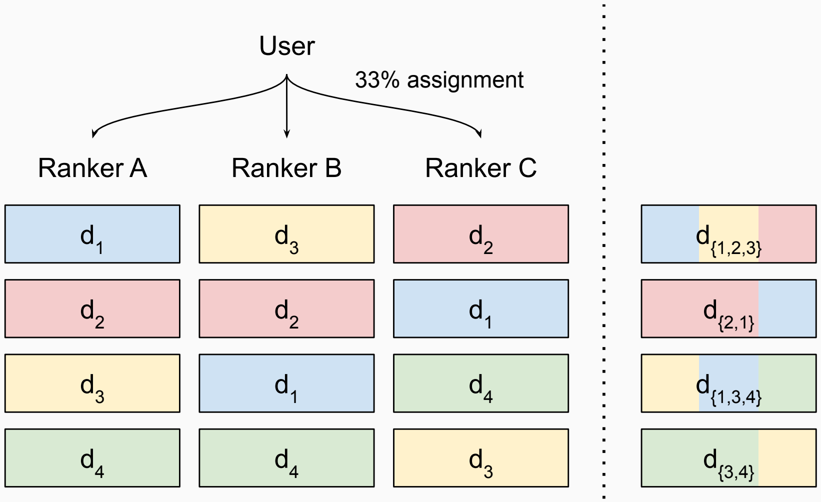
Intervention Harvesting

- **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. (2019b))

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. (2018a)) or a **Dual Learning Objective** (Ai et al. (2018)).

Conclusion

Conclusion

In this lecture we discussed:

- **User-interactions** on rankings are **very biased**.
- **Counterfactual Learning to Rank:**
  - Unbiased learning from historical interaction logs.
  - Correct for position bias with inverse propensity scoring.
  - Requires an explicit user model.
- Estimating **users' examination probabilities:**
  - Through randomization or joint learning.

Future Directions

Future Directions

- **Unbiased Learning to Rank for:**
  - Recommender systems (Schnabel et al., 2016).
  - Personalized rankings in search or recommendation.
- **Correcting for more biases:**
  - Presentation bias, a well known but unaddressed bias.
  - Social biases (fair/ethical A.I.) especially when ranking people.
- **Learning from other signals:**
  - Likes, dwell time, conversion, purchases, watch-time, etc.

This is an extremely active area of research!

Questions and Answers

Thank you for participating!

Notation

Notation Used in the Slides i

Definition	Notation	Example
Query	q	–
Candidate documents	D	–
Document	$d \in D$	–
Ranking	R	$(R_1, R_2, \ldots, R_n)$
Document at rank i	R_i	$R_i = d$
Relevance	$y : D \to \mathbb{N}$	$y(d) = 2$
Ranker model with weights θ	$f_\theta : D \to \mathbb{R}$	$f_\theta(d) = 0.75$
Click	$c_i \in \{0, 1\}$	–
Observation	$o_i \in \{0, 1\}$	–
Rank of d when f_θ ranks D	$\mathit{rank}(d \mid f_\theta, D)$	$\mathit{rank}(d \mid f_\theta, D) = 4$

Notation Used in the Slides ii

Differentiable upper bound on $\mathit{rank}(d, \mid f_\theta, D)$	$\overline{\mathit{rank}}(d, \mid f_\theta, D)$	–
Average Relevant Position metric	*ARP*	–
Discounted Cumulative Gain metric	*DCG*	–
Precision at k metric	*Prec*@k	–
A performance measure or estimator	Δ	–

Resources i

- Tensorflow Learning to Rank, allows for inverse propensity scoring: `https://github.com/tensorflow/ranking`
- Inverse Propensity Scored Rank-SVM: `https://www.cs.cornell.edu/people/tj/svm_light/svm_proprank.html`
- Data and code for comparing counterfactual and online learning to rank `http://github.com/rjagerman/sigir2019-user-interactions`
- An older online learning to rank framework: Lerot `https://bitbucket.org/ilps/lerot/`

References i

A. Agarwal, S. Basu, T. Schnabel, and T. Joachims. Effective evaluation using logged bandit feedback from multiple loggers. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 687–696. ACM, 2017.

A. Agarwal, K. Takatsu, I. Zaitsev, and T. Joachims. A general framework for counterfactual learning-to-rank. In *42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*, page (to appear). ACM, 2019a.

A. Agarwal, I. Zaitsev, X. Wang, C. Li, M. Najork, and T. Joachims. Estimating position bias without intrusive interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 474–482, 2019b.

Q. Ai, K. Bi, C. Luo, J. Guo, and W. B. Croft. Unbiased learning to rank with unbiased propensity estimation. In *Proceedings of the 41st International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 385–394. ACM, 2018.

C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

References ii

O. Chapelle and Y. Chang. Yahoo! Learning to Rank Challenge Overview. *Journal of Machine Learning Research*, 14:1–24, 2011.

Z. Fang, A. Agarwal, and T. Joachims. Intervention harvesting for context-dependent examination-bias estimation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 825–834, 2019.

R. Jagerman, H. Oosterhuis, and M. de Rijke. To model or to intervene: A comparison of counterfactual and online learning to rank from user interactions. In *42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*, page (to appear). ACM, 2019.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

References iii

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

D. Lefortier, P. Serdyukov, and M. de Rijke. Online exploration for detecting shifts in fresh intent. In *CIKM 2014: 23rd ACM Conference on Information and Knowledge Management*. ACM, November 2014.

T. Qin and T.-Y. Liu. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.

M. Sanderson. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375, 2010.

T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims. Recommendations as treatments: debiasing learning and evaluation. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pages 1670–1679. JMLR. org, 2016.

References iv

X. Wang, M. Bendersky, D. Metzler, and M. Najork. 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*, pages 115–124. ACM, 2016.

X. Wang, N. Golbandi, M. Bendersky, D. Metzler, and M. Najork. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 610–618. ACM, 2018a.

X. Wang, C. Li, N. Golbandi, M. Bendersky, and M. Najork. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1313–1322. ACM, 2018b.

Acknowledgments

All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their employers and/or sponsors.