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

Who are we?

Harrie Oosterhuis
PhD student at
U. Amsterdam

Rolf Jagerman
PhD student at
U. Amsterdam

Maarten de Rijke
Professor at
U. Amsterdam

Learning goals

At the end of this part of the tutorial, you should:

- be convinced of the **importance of learning to rank from user interactions**
- understand the most relevant algorithms in **counterfactual/online learning to rank**
- be capable of **deciding which type of learning to rank from user interaction methods to use in which cases**
- be able to **contribute to further development** of learning to rank from user interactions.

What are we going to do?

Part 1: Introduction

Part 2: Counterfactual Learning to Rank

Part 3: Online Learning to Rank

Part 4: Conclusion

Online Audience Questions

If you have any questions we want to hear them!

Tweet us at: @mdr @RolfJagerman or @HarrieOos and use #ltrtutorial

We will address your questions in a follow-up video.

Part 1: Introduction

Part 1: Introduction

This part will cover the following topics:

- **Supervised learning to rank** from annotations.
- **Limitations** of learning to rank from annotated datasets
- Learning from user interactions
  - Noise and bias.

Learning to Rank

Learning to Rank

Learning to Rank (LTR) is:

> "... the task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance."
>
> — Liu et al. (2009)

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

- Key component for **search**, **recommendation**, and **digital assistants**.

Learning to Rank: Problem Definition

The **ranking** R of **ranker** f_θ over a document set D is:

$$R = (R_1, R_2, R_3, \dots)$$

where documents are **ordered** by their (descending) **scores**:

$$f_\theta(R_1) \geq f_\theta(R_2) \geq f_\theta(R_3) \geq \dots,$$

and **every document** is in the ranking:

$$d \in D \iff d \in R.$$

Learning to Rank: Problem Definition

For this tutorial, we will cast **the goal of LTR** as:

- Find the **parameters** θ for the **model** f_θ,
  where sorting documents d according to their scores $f_\theta(d)$
  results in the **most optimal rankings**.

We will later define what is *optimal* according to well-known ranking metrics.

Supervised Learning to Rank

Supervised Learning to Rank: Setup

Supervised LTR methods require supervision from **annotated datasets**, these contain:

- **Queries**, representing queries users will issue,
- **Documents**, per query a preselected set of documents to be ranked,
- **Relevance Labels**, indicating relevance/preference per document-query pair.

Supervised LTR methods are commonly divided in **three groups**:

- Pointwise, Pairwise, and Listwise.

Supervised Learning to Rank: Annotations

Relevance labels are gathered by **human judges** that annotate document-query pairs. The resulting labels can be used for **supervision** during learning (Chapelle and Chang, 2011; Liu et al., 2007).

Let $y(d)$ indicate the **relevance** of document d to the current query,
in this tutorial we never talk about the same document w.r.t. to multiple queries,
thus we can keep the query out of the notation $y(d)$.

Pointwise Methods

The pointwise approach casts LTR as the standard machine learning task of **label prediction**, by using a classification or regression loss.

For instance, the mean squared error is a common **regression loss**:

$$\mathcal{L}_{\mathit{pointwise}} = \frac{1}{N} \sum_{i=1}^{N} \big(f_\theta(d_i) - y(d_i)\big)^2.$$

Pointwise Methods: Problem

The **issue** with pointwise methods is that they **ignore** that model **scores** are **used to rank** documents.

In other words:

- A pointwise loss only wants scores to be **close to the labels**,
- LTR only wants scores to result in the **correct ordering**.

In practice, pointwise methods **compromise ordering** to get scores closer to the labels.

Pairwise Methods

The pairwise approach realizes that **ordering** is based on **relative score differences**. It uses a loss based on **pairs of documents** with a difference in relevance (Joachims, 2002).

For instance, an (unnormalized) **pairwise hinge-loss**:

$$\mathcal{L}_{\textit{pairwise}} = \sum_{y(d_i) > y(d_j)} \max\Big(0,\, 1 - \big(f_\theta(d_i) - f_\theta(d_j)\big)\Big).$$

Pairwise Methods: Problem

The **problem** with the pairwise approach:

- **every document pair** is treated as **equally important**,
- often **users care** more about the **top-10** than the **top-100**.

Thus pairwise methods may compromise quality in the top-10 to improve the ordering in the tail of the top-100.

Listwise Methods

The idea of **listwise methods** is to **optimize ranking metrics** directly. However, metrics based on the rank function are **not differentiable**.

For instance, the **Discounted Cumulative Gain** metric:

$$\mathsf{DCG} = \sum_{i=1}^{N} \frac{y(d_i)}{\log_2(\mathsf{rank}(d_i) + 1)}.$$

Problem: $\log_2(\mathsf{rank}(d_i) + 1)$ is not differentiable.

Listwise Methods: Examples

Solutions to this problem:

- use **probabilistic approximations** of ranking:
  e.g. ListNet (Cao et al., 2007), ListMLE (Xia et al., 2008),
- use **heuristics** or **bounds** on metrics:
  e.g. LambdaRank (Burges, 2010), LambdaLoss (Wang et al., 2018c).

For instance, the **LambdaRank** loss is a proven bound on DCG:

$$\mathcal{L}_{\mathit{LambdaRank}} = \sum_{y(d_i)>y(d_j)} \log\left(1 + e^{f_\theta(d_j)-f_\theta(d_i)}\right) \cdot |\Delta \mathrm{DCG}|.$$

Limitations of Annotated Datasets

Learning to Rank from Annotated Datasets

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

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

However, over time **several limitations** of this approach have become apparent.

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., 2016a).
- **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.

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

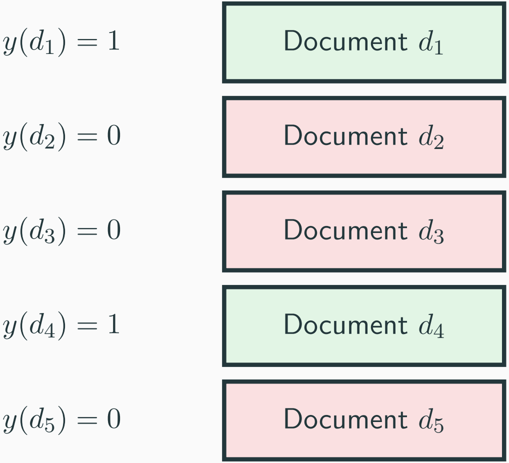
If we **know** the **true relevance labels** ($y(d_i)$ for all i), we can compute any additive linearly decomposable IR metric as:

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

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

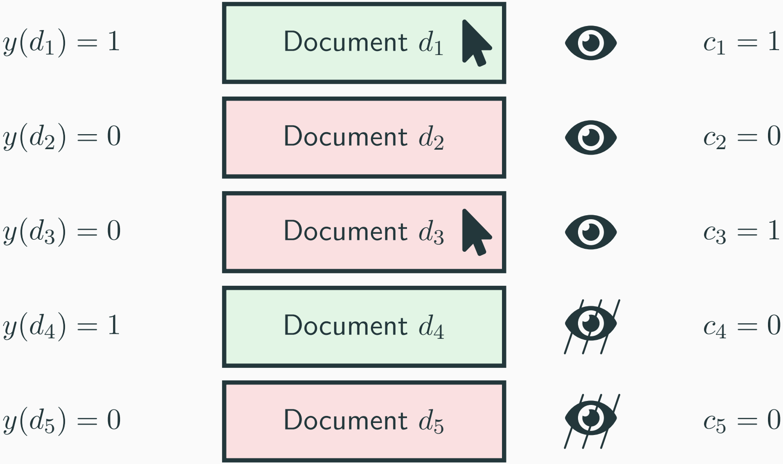
$y(d_5) = 0$ Document d_5

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)**:

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

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**:

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

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

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

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**:

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

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

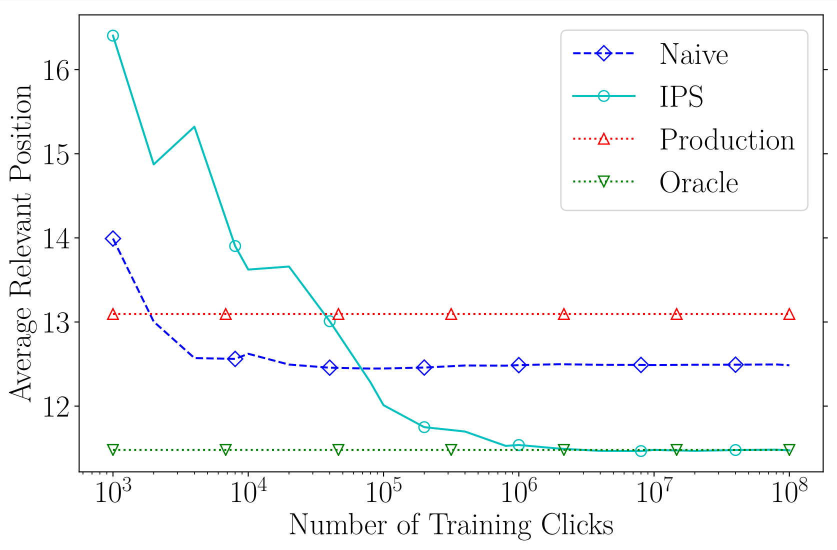
Propensity-weighted LTR: Semi-synthetic Experiments

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., $P(o_i = 1 \mid R, d_i) = \frac{1}{\textit{rank}(d \mid R)}$.
- 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:

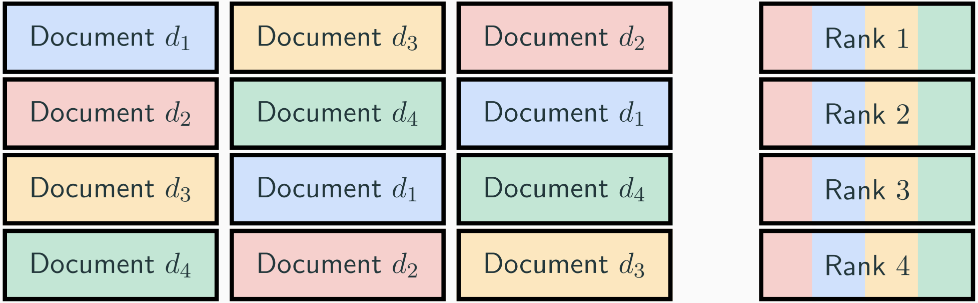
$$P(o_i = 1 \mid i).$$

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.

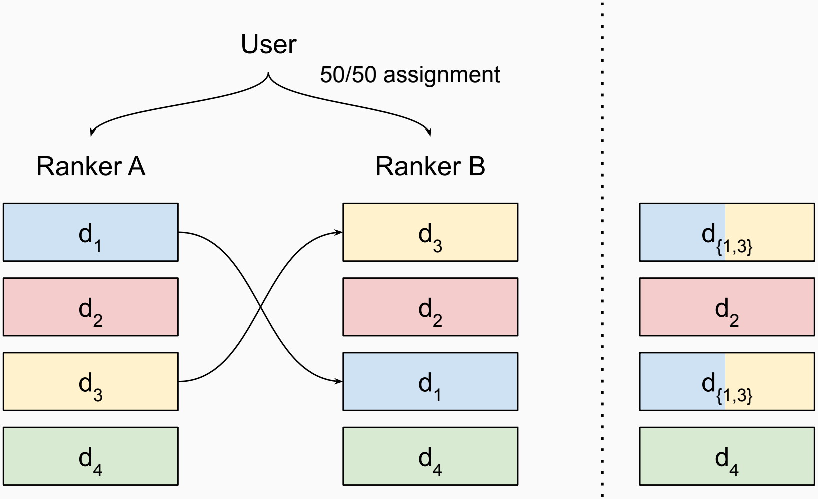
There are various methods that minimize the impact to the user:

- **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).

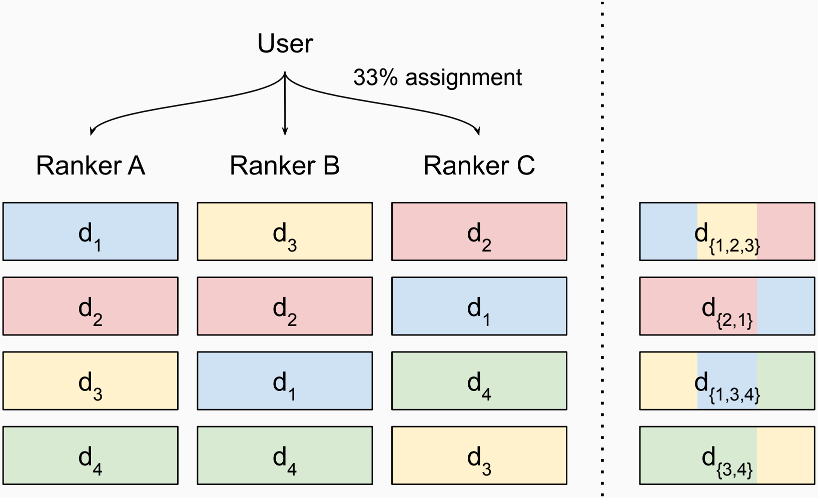
Intervention Harvesting

- 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



User

33% assignment

Ranker A	Ranker B	Ranker C
d_1	d_3	d_2
d_2	d_2	d_1
d_3	d_1	d_4
d_4	d_4	d_3

$d_{\{1,2,3\}}$

$d_{\{2,1\}}$

$d_{\{1,3,4\}}$

$d_{\{3,4\}}$

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*$(d_i, R) = k$:

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

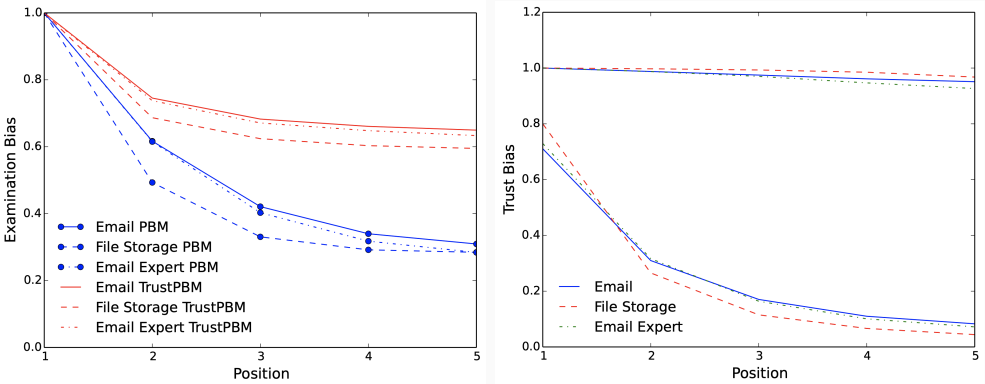
Correcting for Trust Bias

The new estimator becomes:

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

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**.

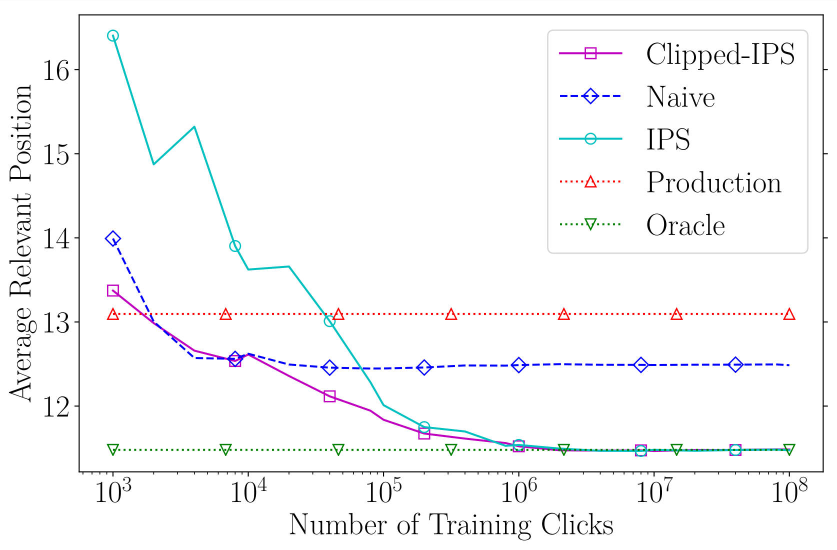
Propensity clipping: Bound the *propensity*, to prevent any single sample from overpowering the rest of the data set:

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

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.

Related Work: Click Models

Click Models: Introduction

Click models form an established branch of IR research also **related** to learning from **user interactions** (Chuklin et al., 2015).

The main goal of click models is:

- find a **model** that **realistically simulates user behavior**.

In practical terms this often means:

- learn to **predict future interactions** of users.

Click Models: Rank-Biased Model

To realistically simulate user behavior click models have to **capture** the **factors** that **influence interactions**. This includes:

- Estimates of the **attractiveness** of documents **(relevance)**.
- **Other factors** that cause/prevent clicks **(biases)**.

Click models can be used for **bias estimation** for **counterfactual LTR**,
i.e., previously discussed EM-based estimation is essentially the Position Biased Model by Craswell et al. (2008).

Click Models: Relevance Modelling

Click models often **separately estimate attractiveness**: $P(c_i = 1 \mid o_i = 1, y(d_i))$, based on document **features** x_i, i.e. with a **learned function**: $m(x_i) \in [0, 1]$.

In theory m can be used for **unbiased evaluation**,
in practice estimating relevance from features x_i is not **accurate enough**.

In contrast, **counterfactual evaluation** does not rely on features, but uses a **debiased frequency-based estimate**:

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

Comparison to Click Models

Click Models:

- aims to **realistically simulate** user behavior.
- models attractiveness and biases.
- can be used for **feature-based evaluation**.

Counterfactual LTR:

- considers the **effect of biases** on the learning **objective**.
- can be used for **unbiased frequency-based evaluation/learning**.

Concluding Part 1 & 2

Conclusion of Part 2

So far we discussed:

- **User interactions** with rankings are **very biased**.
- **Counterfactual Learning to Rank:**
  - Correct for position bias with inverse propensity scoring.
  - Requires an explicit user model.
- Unbiased learning from historical interaction logs.

In the next two parts we will look at:

- **Online Learning to Rank:**
  - Algorithms that directly interact with users.
  - Handle biases through randomization.
- A **comparison** of both methodologies.

Part 3: Online Learning to Rank

Online Learning to Rank: Overview

This part will cover the following topics:

- **Online Evaluation**
    - Comparing rankers through interleaving.
- **Dueling Bandit Gradient Descent**
    - Learning to rank as an interactive dueling bandit problem.
- **Pairwise Differentiable Gradient Descent**
    - Learning to rank through unbiased pairwise optimization.
- **Comparison of PDGD and DBGD**
    - Theoretical differences and empirical comparisons.

Related Work: Bandits for Ranking

Ranking as a K-Armed Bandit

In the past, ranking has been modelled as a K-armed bandit (Busa-Fekete and Hüllermeier, 2014).

These methods aim to find the **optimal ranking for a single query**.

Ranking bandit methods include:

- **Upper confidence bounds** on relevances per document (Kveton et al., 2015).
- **Divide and conquer**: split documents in groups so that there are high-confidence relevance differences between groups (Lattimore et al., 2018).
- **Click-through-rate estimation** per document similar to counterfactual LTR (Lagrée et al., 2016).

Ranking Bandits and Learning to Rank

The goal of ranking bandit algorithms is:

- the **optimal ranking** for a **single query**.

The results from these algorithms do **not generalize** to other queries, i.e., there is **no resulting ranking model**.

Advantage: rankings **not limited by features** (Zoghi et al., 2016).
Disadvantage: **learning from scratch** for every new query.

Very different from the **goal** of LTR as defined for this **tutorial**:

- to find a **ranking model** that **generalizes** well across user queries.

Online Evaluation

Online Evaluation: Introduction

We have seen:

- Counterfactual evaluation corrects for position bias in historical logs by explicitly modelling the user's examination probabilities.

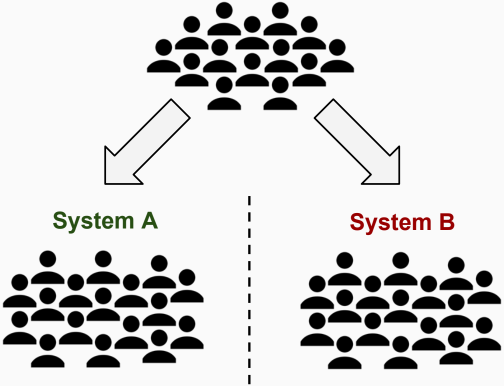
One way of getting these **explicit probabilities** is through **randomization**.

Alternatively, older methods use **randomization** to **directly perform evaluation**:

- A/B testing
- Interleaving

They answer the **question**: **Should ranker A be preferred over ranker B?**

Online Evaluation: A/B testing



A/B testing **randomizes system exposure to users** to measure differences.

Online Evaluation: Interleaving

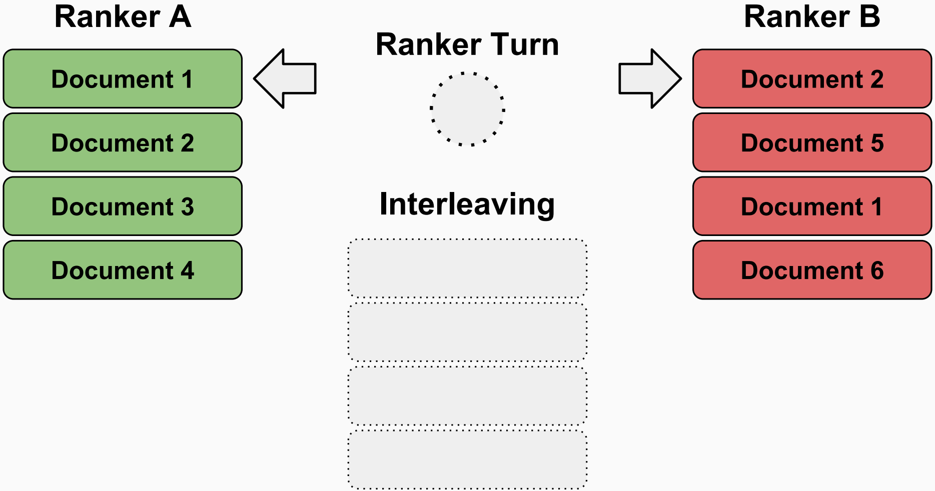
A/B testing is powerful and widely applicable, it is **not specific for rankings**.

Specific aspects of interactions in rankings can be used for **more efficient comparisons**.

Interleaving (Joachims, 2003):

- Take the two rankings for a query from two rankers .
- Create an **interleaved ranking**, based on both rankings.
- **Clicks** on an interleaved ranking provide **preference signals** between rankers.

Online Evaluation: Team-Draft Interleaving



Ranker A

Document 1

Document 2

Document 3

Document 4

Ranker Turn

Interleaving

Ranker B

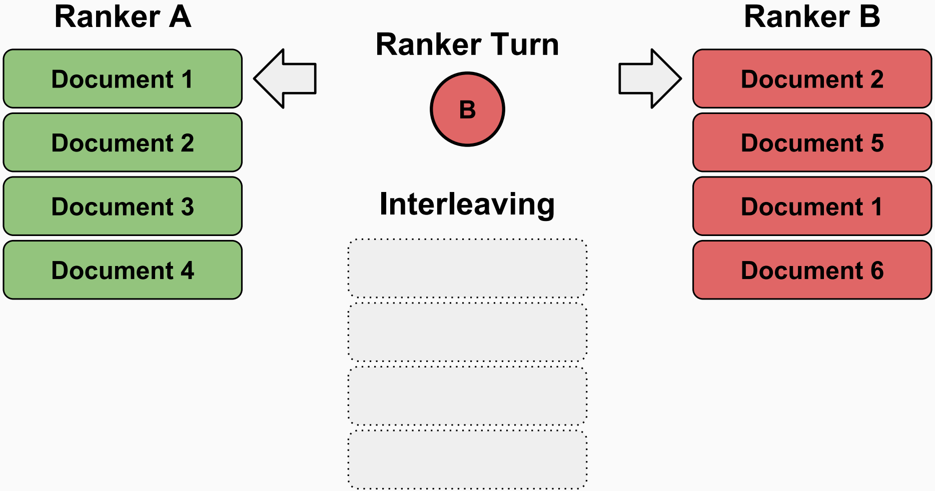
Document 2

Document 5

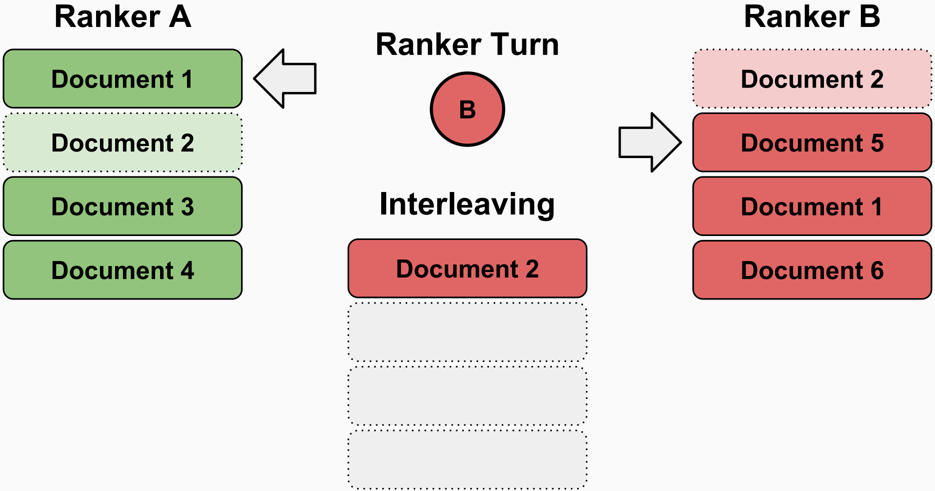
Document 1

Document 6

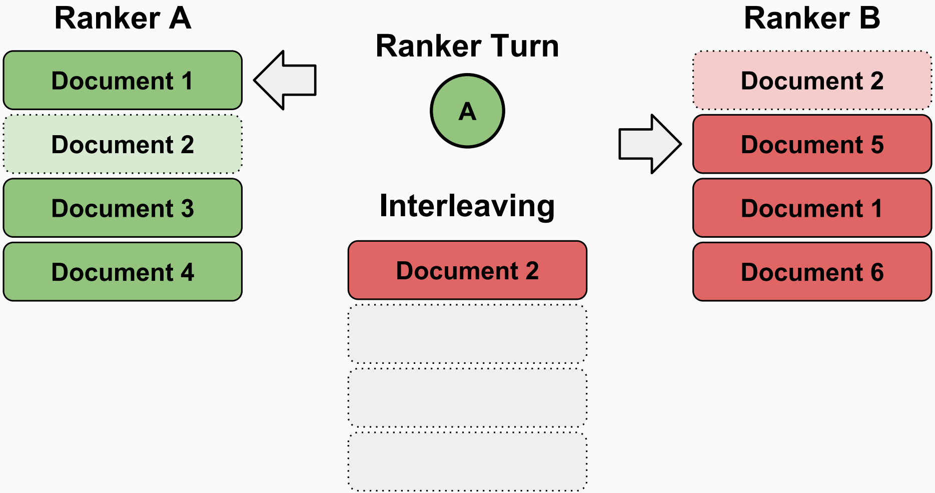
Online Evaluation: Team-Draft Interleaving



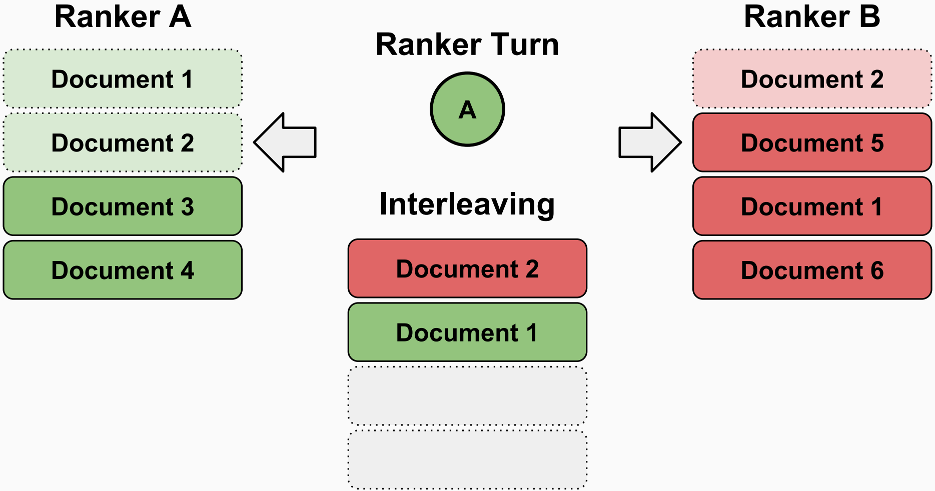
Online Evaluation: Team-Draft Interleaving



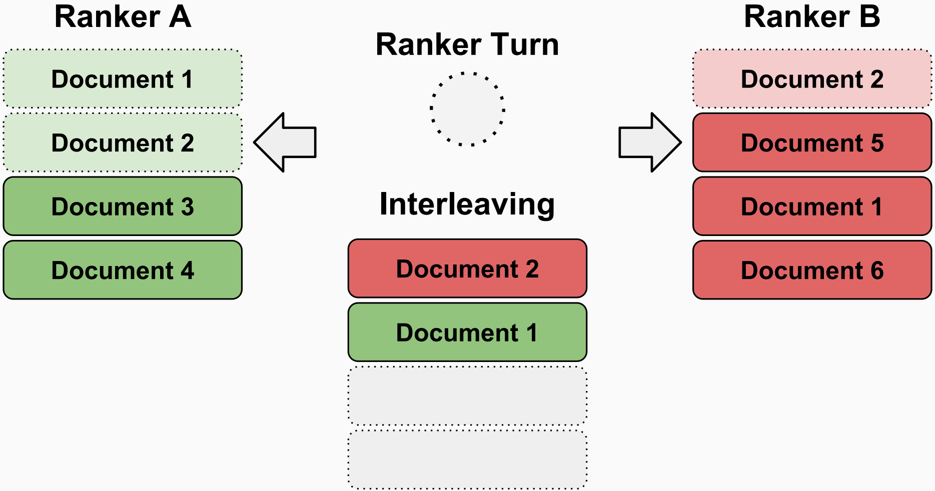
Online Evaluation: Team-Draft Interleaving



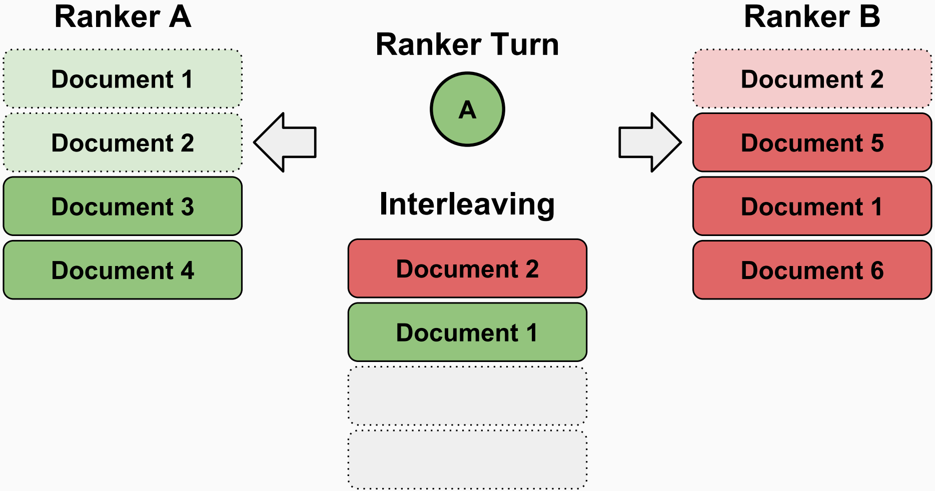
Online Evaluation: Team-Draft Interleaving



Online Evaluation: Team-Draft Interleaving



Online Evaluation: Team-Draft Interleaving



Ranker A

Document 1

Document 2

Document 3

Document 4

Ranker Turn

A

Interleaving

Document 2

Document 1

Ranker B

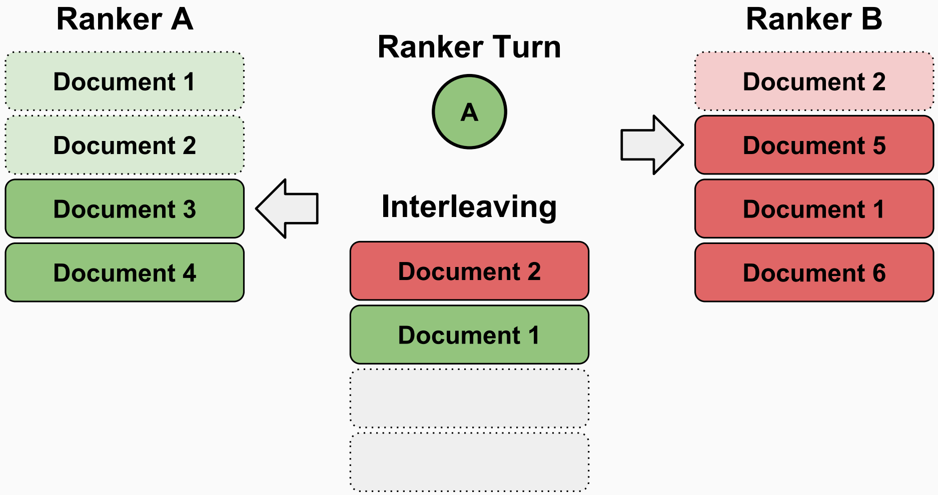
Document 2

Document 5

Document 1

Document 6

Online Evaluation: Team-Draft Interleaving



Ranker A

Document 1

Document 2

Document 3

Document 4

Ranker Turn

A

Interleaving

Document 2

Document 1

Ranker B

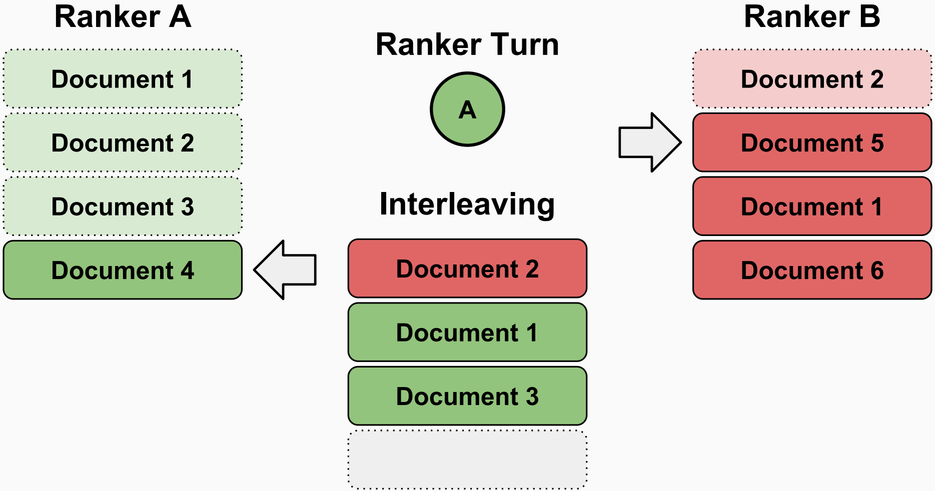
Document 2

Document 5

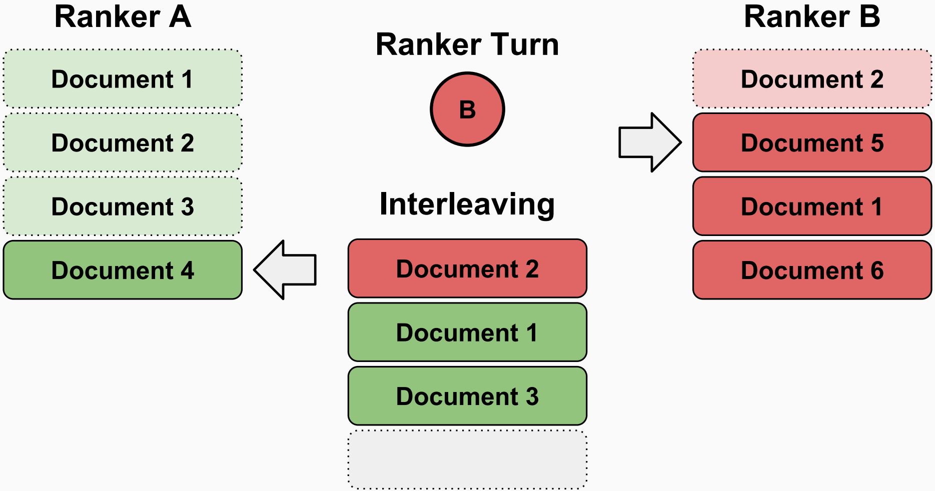
Document 1

Document 6

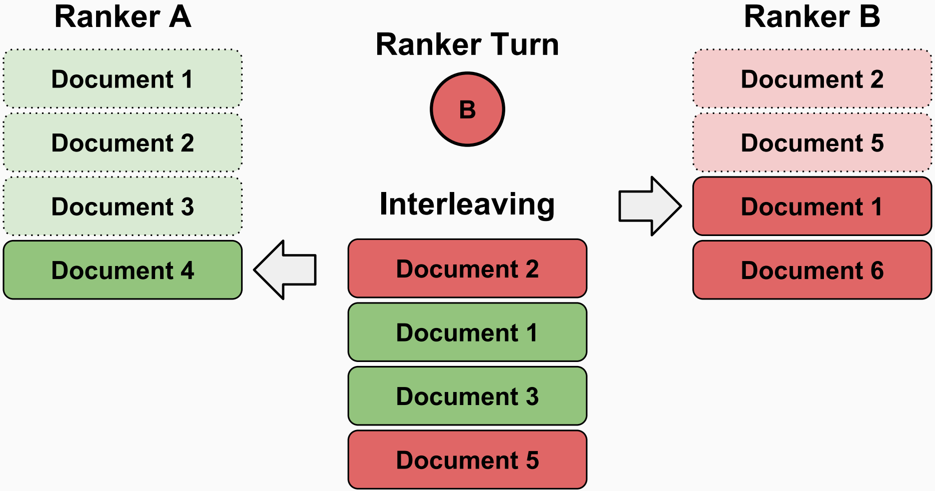
Online Evaluation: Team-Draft Interleaving



Online Evaluation: Team-Draft Interleaving



Online Evaluation: Team-Draft Interleaving



Ranker A

Document 1

Document 2

Document 3

Document 4

Ranker Turn

B

Interleaving

Document 2

Document 1

Document 3

Document 5

Ranker B

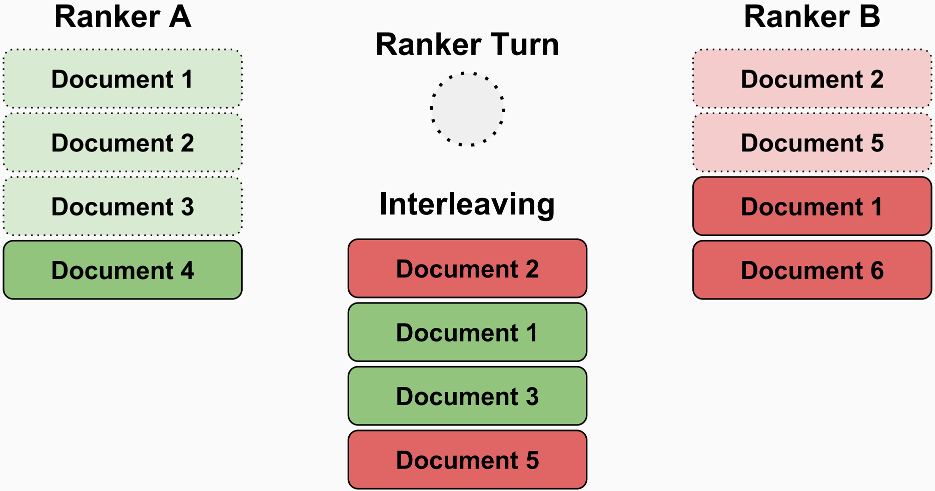
Document 2

Document 5

Document 1

Document 6

Online Evaluation: Team-Draft Interleaving



Ranker A

Document 1

Document 2

Document 3

Document 4

Ranker Turn

Interleaving

Document 2

Document 1

Document 3

Document 5

Ranker B

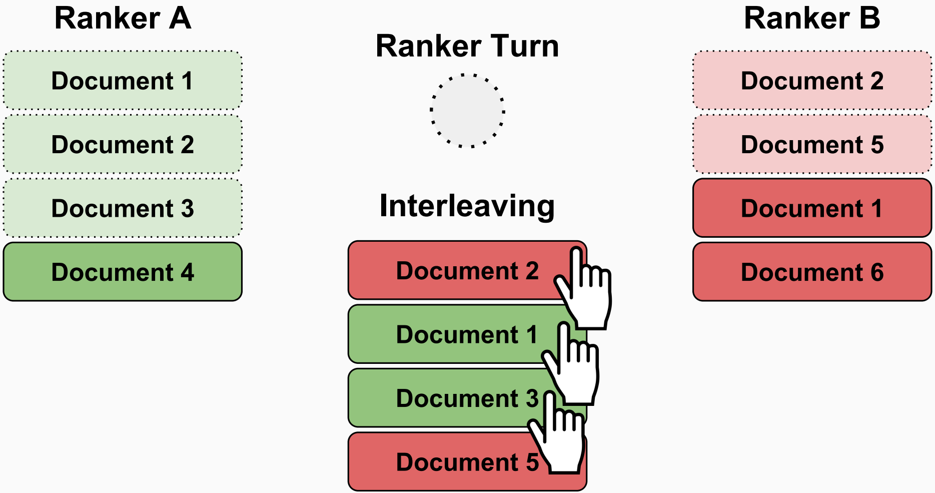
Document 2

Document 5

Document 1

Document 6

Online Evaluation: Team-Draft Interleaving



Ranker A

Document 1

Document 2

Document 3

Document 4

Ranker Turn

Interleaving

Document 2

Document 1

Document 3

Document 5

Ranker B

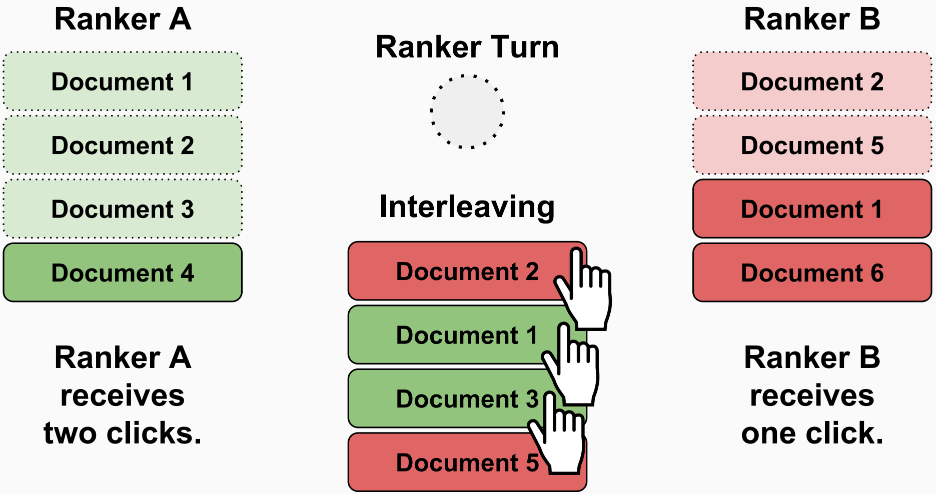
Document 2

Document 5

Document 1

Document 6

Online Evaluation: Team-Draft Interleaving



Ranker A

Document 1

Document 2

Document 3

Document 4

Ranker Turn

Interleaving

Document 2

Document 1

Document 3

Document 5

Ranker B

Document 2

Document 5

Document 1

Document 6

Ranker A receives two clicks.

Ranker B receives one click.

Online Evaluation: Interleaving

The idea behind interleaving:

- **Randomize display positions** of documents to deal with position bias.
- Limit randomization to **maintain user experience**.

Team-Draft Interleaving (Radlinski et al., 2008) is **affected by position bias**:

- Similar rankers can be inferred equal when a preference should be found.

Other interleaving methods are **proven** to be **unbiased**[1]:

- **Probabilistic Interleaving** (Hofmann et al., 2011)
- **Optimized Interleaving** (Radlinski and Craswell, 2013)

[1] Different definition of unbiased than the first part of this tutorial.

Online Evaluation: Interleaving

Interleaving requires **magnitudes fewer interactions** for a reliable preference than A/B testing (Chapelle et al., 2012; Yue et al., 2010).

Unlike counterfactual evaluation, interleaving **is interactive**.

- It is not effective on historical data (Hofmann et al., 2013).

Efficiency comes from:

- displaying the **most important documents** first,
- and looking for **relative differences**.

Providing a reliable, efficient and interactive evaluation methodology.

Dueling Bandit Gradient Descent

Dueling Bandit Gradient Descent: Introduction

Introduced by Yue and Joachims (2009) as the **first online learning to rank** method.

Intuition:

- if **online evaluation** can tell us if a **ranker is better** than another, then we can use it to **find an improvement** of our system.

By **sampling model variants** and **comparing** them with **interleaving**, the *gradient* of a model w.r.t. user satisfaction can be **estimated**.

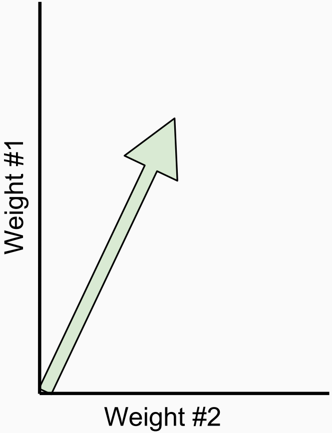
Dueling Bandit Gradient Descent: Method

Start with the **current** ranking model **parameters**: θ_b.

Then indefinitely:

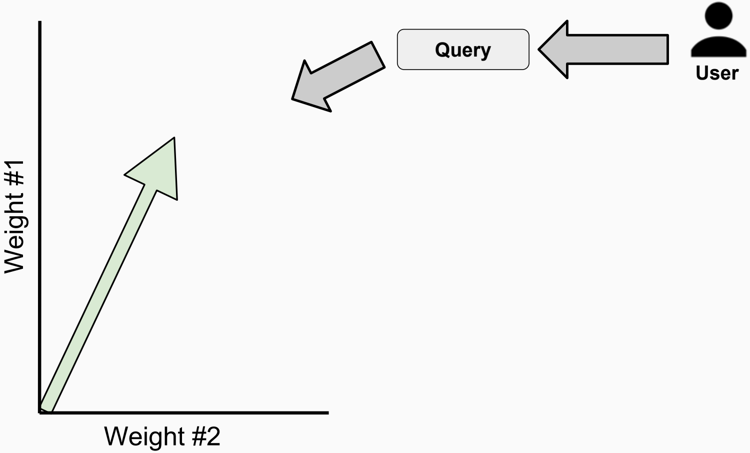
1. Wait for a user query.
2. **Sample a random direction** from the unit sphere: u, (thus $|u| = 1$).
3. Compute the **candidate ranking model** $\theta_c = \theta_b + u$, (thus $|\theta_b - \theta_c| = 1$).
4. Get the **rankings** of θ_b and θ_c.
5. **Compare** θ_b and θ_c using interleaving.
6. If θ_c wins the **comparison**:
    - **Update** the current model: $\theta_b \leftarrow \theta_b + \eta(\theta_c - \theta_b)$

Dueling Bandit Gradient Descent: Visualization

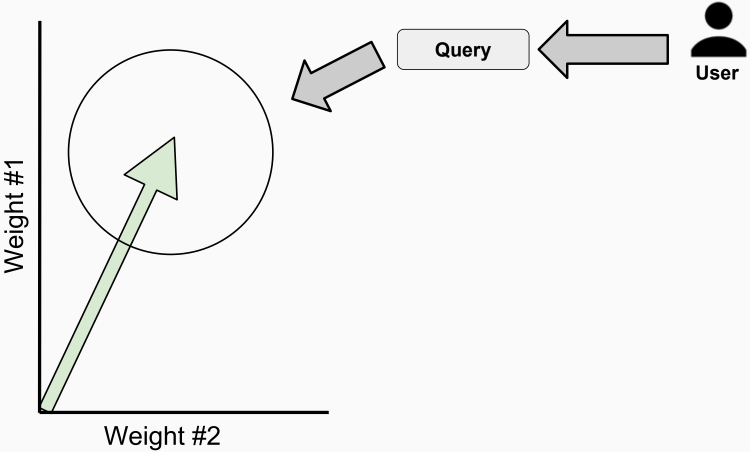


User

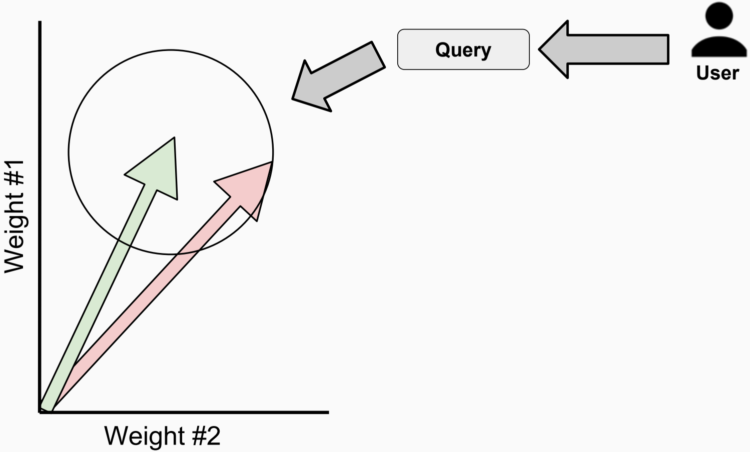
Dueling Bandit Gradient Descent: Visualization



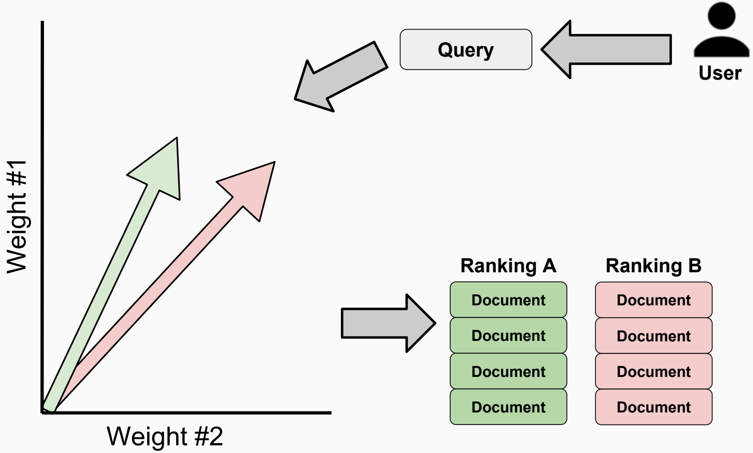
Dueling Bandit Gradient Descent: Visualization



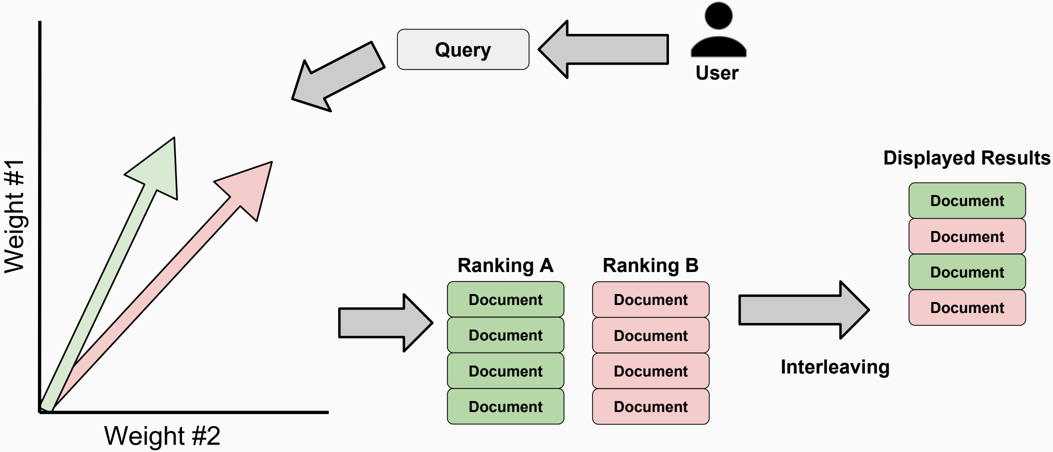
Dueling Bandit Gradient Descent: Visualization



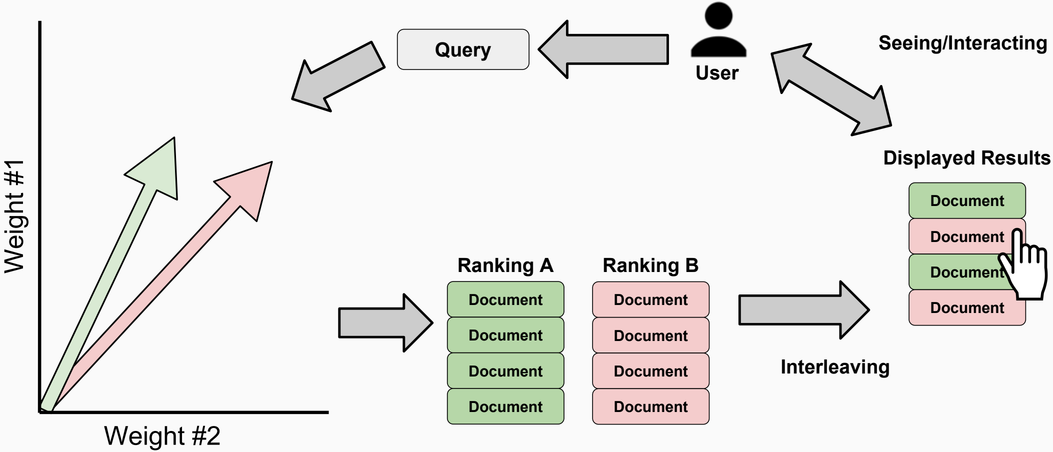
Dueling Bandit Gradient Descent: Visualization



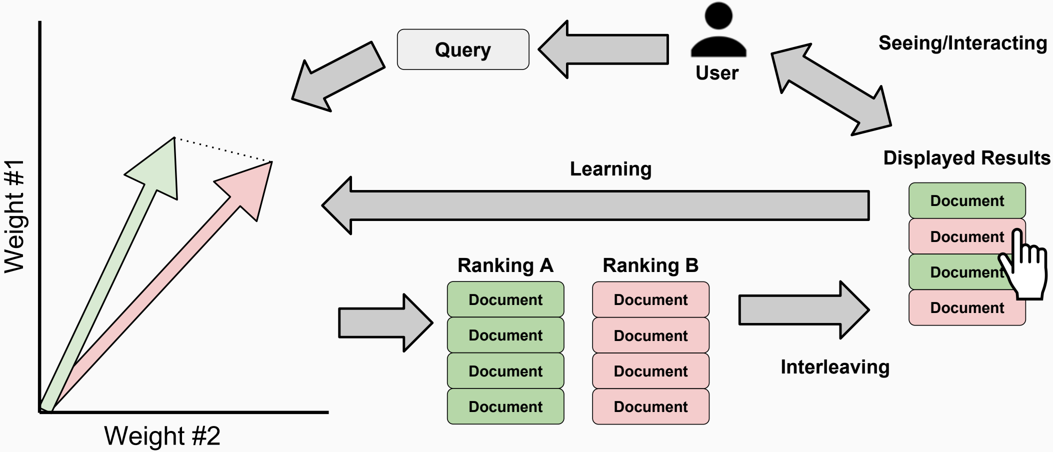
Dueling Bandit Gradient Descent: Visualization



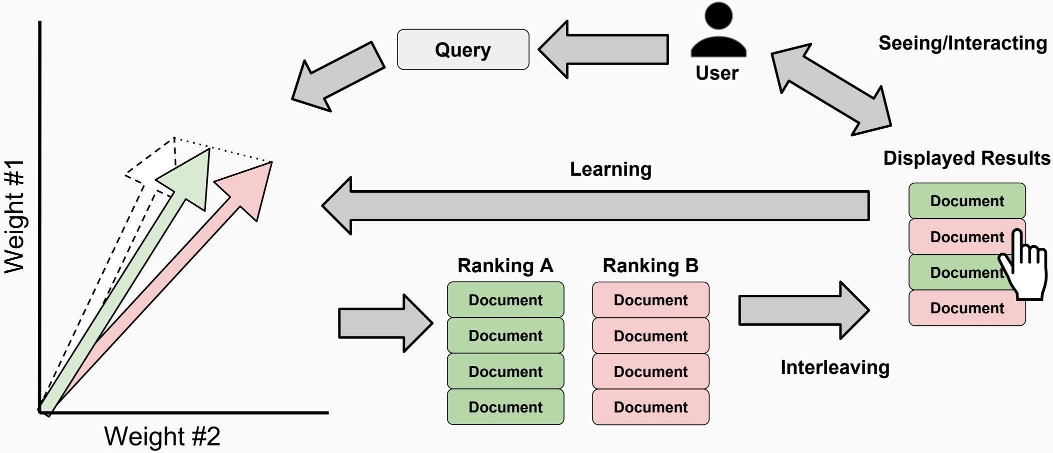
Dueling Bandit Gradient Descent: Visualization



Dueling Bandit Gradient Descent: Visualization



Dueling Bandit Gradient Descent: Visualization



Dueling Bandit Gradient Descent: Properties

Yue and Joachims (2009) prove that under the **assumptions**:

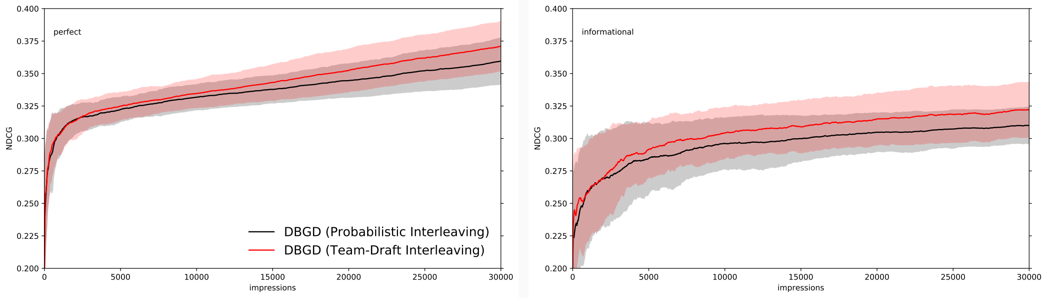
- There is a **single optimal** set of parameters: θ^*.
- The **utility space** w.r.t. θ is **smooth**,
  i.e., small changes in θ lead to small changes in user experience.

Then Dueling Bandit Gradient Descent is **proven** to have a **sublinear regret**:

- The algorithm will **eventually** approximate the ideal model.
- The duration of time is effected by the number of parameters of the model, the smoothness of the space, the unit chosen, etc.

Dueling Bandit Gradient Descent: Visualization

Simulations based on offline datasets: **user behavior** is based on the **annotations**.
As a result, we can **measure** how close the **model** is getting to their **satisfaction**.



**Simulated results on the MSLR-WEB10k dataset,
a perfect user (left) and an informational user (right).**

Image credits: (Oosterhuis, 2018).

Reusing Historical Interactions

Reusing Historical Interactions

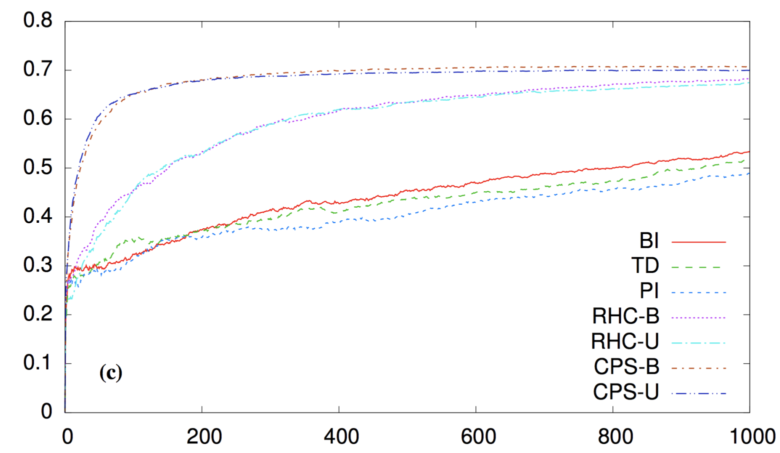
Hofmann et al. (2013) introduced the idea of **guiding exploration** by **reusing previous interactions**.

Intuition: if **previous interactions** showed that a **direction is unfruitful** then we should **avoid it in the future**.

Candidate Pre-Selection:

- Sample a **large number** of rankers to create a **candidate set**.
- **Compare two** candidate rankers based on a **historical interaction**.
- **Remove loser** from candidate set.
- **Repeat** until a **single candidate** is left.

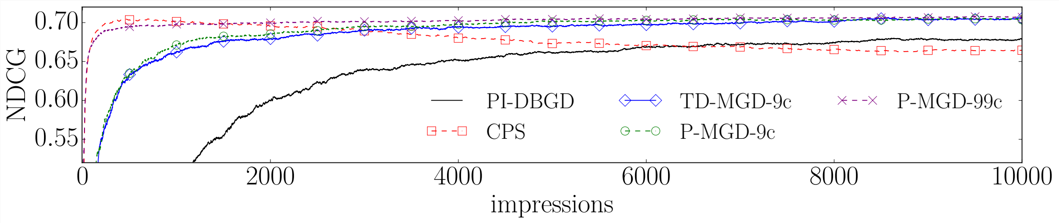
Reusing Historical Interactions: Performance



Simulated results on the NP2003 dataset.

Image credits: graph from (Hofmann et al., 2013).

Reusing Historical Interactions: Long Term Performance



Simulated results on the NP2003 dataset.

Remember, in the online setting the **performance cannot be measured**, thus **early-stopping is unfeasible**.

Image credits: graph from (Oosterhuis et al., 2016).

Reusing Historical Interactions: Other Work

Besides Hofmann et al. (2013) **other work** has also tried **reusing historical interactions** for online learning to rank: (Zhao and King, 2016; Wang et al., 2018a).

The problem with these works is that:

- they **do not consider the long-term convergence**.
- they were **not evaluated** on the **largest available industry datasets**.

As a result, it is **still unclear** whether we can **reliably reuse historical interactions** during online learning.

Multileave Gradient Descent

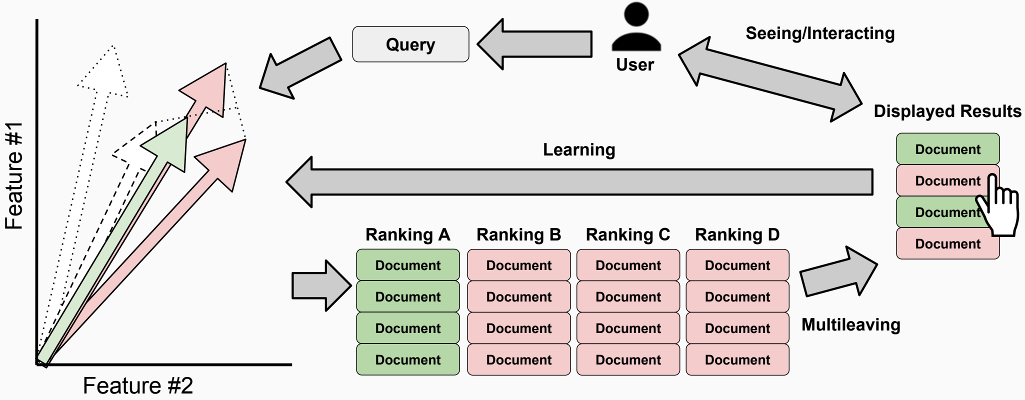
Multileave Gradient Descent

The introduction of **multileaving** in online evaluation allowed for **multiple rankers being compared simultaneously** from a single interaction.

A **natural extension** of Dueling Bandit Gradient Descent is to combine it with multileaving, resulting in **Multileave Gradient Descent** (Schuth et al., 2016).

Multileaving allows comparisons with **multiple candidate rankers**, **increasing** the **chance of finding an improvement**.

Multileave Gradient Descent: Visualization



Multileave Gradient Descent: Results

Results on the MSRL10k dataset under simulated users:

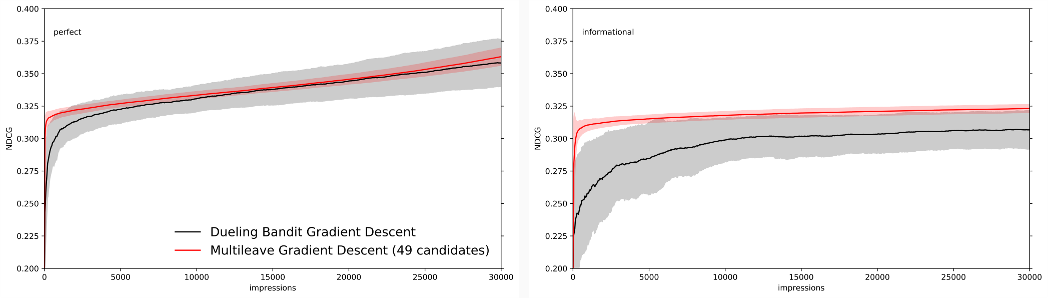


Image credits: (Oosterhuis, 2018).

Multileave Gradient Descent: Conclusion

Properties of Multileave Gradient Descent:

- **Vastly speeds up the learning rate** of Dueling Bandit Gradient Descent.
  - Much better user experience.
- Instead of **limiting (guiding) exploration**, it is done more **efficiently**.
- **Huge computational costs**, large number of rankers have to be applied.

Problems with Dueling Bandit Gradient Descent

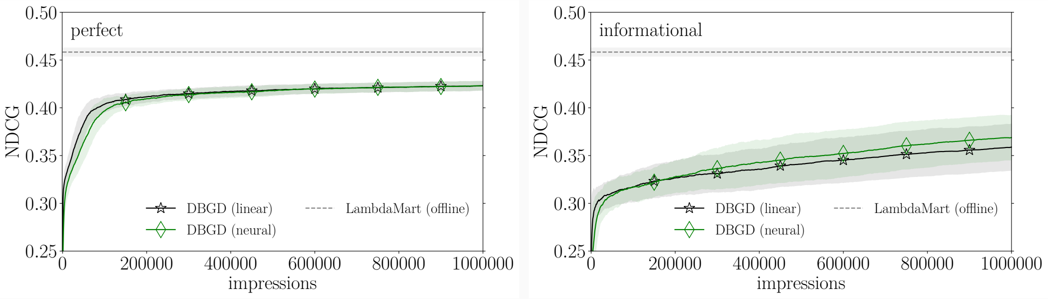
Problems with Dueling Bandit Gradient Descent

A **problem** with Dueling Bandit Gradient Descent and **all its extensions**:

- Their **performance at convergence** is **much worse** than offline approaches, even **under ideal user interactions**.

DBGD problems: Empirical

Results on the MSRL10k dataset under simulated users:



How is this possible, if it has **proven sub-linear regret**?

Image credits: (Oosterhuis, 2018).

Problems with the Dueling Bandit Gradient Descent Bounds

Remember the **regret** of Dueling Bandit Gradient Descent made **two assumptions**:

- There is a **single optimal model**: θ^*.
- The **utility space is smooth** w.r.t. to the model weights θ.

These **assumptions do not hold** for all models that are used in practice (Oosterhuis and de Rijke, 2019).

To prove this we use the fact that **the utility u is scale invariant** w.r.t. a ranking function $f_\theta(\cdot)$:

$$\forall \theta, \quad \forall \alpha \in \mathbb{R}_{>0}, \quad u(f_\theta(\cdot)) = u(\alpha f_\theta(\cdot)).$$

DBGD Assumptions: Single Optimal Model

First assumption: There is a **single optimal model**: θ^*.

For any linear or neural model:

- if θ^* has the **optimal performance**,
- then $\theta' = \alpha\theta$ has the **same performance**, *(linear model)* or multiplying the final weight matrix with α, *(neural model)*.

Therefore, there can **never** be a **single optimal model** θ^*.

DBGD Assumptions: Smoothness

Second assumption: The **utility space is smooth** w.r.t. to the model weights θ:

$$\exists L \in \mathbb{R}, \quad \forall (\theta_a, \theta_b) \in \mathcal{W}, \quad |u(\theta_a) - u(\theta_b)| < L\|\theta_a - \theta_b\|.$$

Since a **linear model** is **scale invariant**:

$$\begin{aligned}\forall \alpha \in \mathbb{R}_{>0}, &\quad |u(\theta_a) - u(\theta_b)| = |u(\alpha\theta_a) - u(\alpha\theta_b)|,\\ \forall \alpha \in \mathbb{R}_{>0}, &\quad \|\alpha\theta_a - \alpha\theta_b\| = \alpha\|\theta_a - \theta_b\|.\end{aligned}$$

Thus the smoothness assumption can be rewritten as:

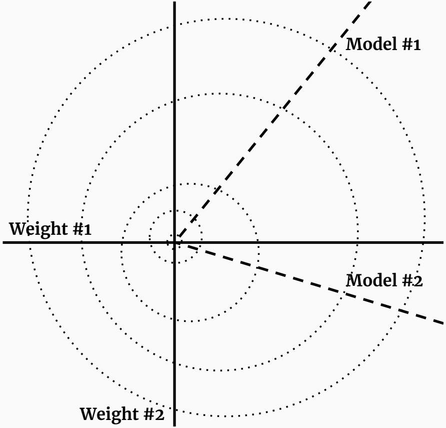
$$\exists L \in \mathbb{R}, \quad \forall \alpha \in \mathbb{R}_{>0}, \quad \forall (\theta_a, \theta_b) \in \mathcal{W}, \quad |u(\theta_a) - u(\theta_b)| < \alpha L\|\theta_a - \theta_b\|.$$

This condition is **impossible to be true** (proof can be extended for neural networks).

DBGD Assumptions: Smoothness Visualization

Intuition behind the **smoothness problem** for linear ranking models:

- **Every model** in a **line** from the origin in any direction is **equivalent**.
- **Any sphere** around the origin contains **every possible ranking model**[a].
- The **distance** between the *best* and the *worst* model becomes **infinitely small** near the origin.



[a] Except for the trivial random model on the origin.

DBGD Problems: Conclusion

Theoretical properties:

- Currently, no **sound regret bounds proven**.

Empirical observations:

- Methods do **not approach optimal performance**.
- Neural models have no advantage over linear models.

Possible solutions:

- Extend the algorithm (the last decade of research) or introduce new model.
- **Find an approach different to the bandit approach.**

Pairwise Differentiable Gradient Descent

Pairwise Differentiable Gradient Descent

We recently introduced **Pairwise Differentiable Gradient Descent** (Oosterhuis and de Rijke, 2018b):

- Very different from previous Online Learning to Rank methods,
  that relied on sampling model variations similar to evolutionary approaches.

Intuition:

- A **pairwise** approach can be made **unbiased**, while being **differentiable**,
  without relying on online evaluation method or the sampling of models.

Plackett Luce Model

Pairwise Differentiable Gradient Descent optimizes a **Plackett Luce** ranking model, this models a **probabilistic distribution over documents**.

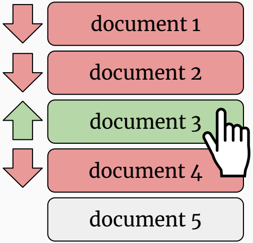
With the ranking scoring model $f_\theta(\mathbf{d})$ the distribution is:

$$P(d|D,\theta) = \frac{\exp^{f_\theta(\mathbf{d})}}{\sum_{d' \in D} \exp^{f_\theta(\mathbf{d'})}}.$$

Confidence is explicitly modelled and **exploration** depends on the **available documents**, thus it **naturally varies per query** and even within the ranking.

Bias in Pairwise Inference

Similar to existing pairwise methods (Oosterhuis and de Rijke, 2017; Joachims, 2002), Pairwise Differentiable Gradient Descent infers **pairwise document preferences from user clicks**:

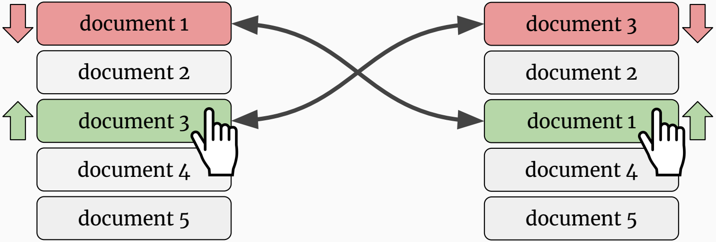


This approach is **biased**:

- Some preferences are **more likely to be inferred** due to **position/selection bias**.

Reversed Pair Rankings

Let $R^*(d_i, d_j, R)$ be R but with the **positions** of d_i and d_j **swapped**:



We assume:

- For a preference $d_i \succ d_j$ inferred from ranking R, if both are **equally relevant** the opposite preference $d_j \succ d_i$ is **equally likely** to be inferred from $R^*(d_i, d_j, R)$.

Then scoring **as if** R and R^* are **equally likely to occur** makes the gradient **unbiased**.

Unbiasing the Pairwise Update

The **ratio** between the probability of the ranking and the reversed pair ranking indicates the **bias between the two directions**:

$$\rho(d_i, d_j, R) = \frac{P(R^*(d_i, d_j, R)|f, D)}{P(R|f, D) + P(R^*(d_i, d_j, R)|f, D)}.$$

We use this ratio to **unbias the gradient estimation**:

$$\nabla f_\theta(\cdot) \approx \sum_{d_i >_{\mathbf{c}} d_j} \rho(d_i, d_j, R) \nabla P(d_i \succ d_j | D, \theta).$$

Unbiasedness of Pairwise Differentiable Gradient Descent

Under the reversed pair ranking assumption, we prove that **the expected estimated gradient** can be written as:

$$E[\nabla f_\theta(\cdot)] = \sum_{d_i,d_j} \alpha_{ij}(f'_\theta(\mathbf{d_i}) - f'_\theta(\mathbf{d_j})).$$

Where the weights α_{ij} will **match the user preferences** in expectation:

$$\begin{aligned} d_i &=_{rel} d_j \Leftrightarrow \alpha_{ij} = 0, \\ d_i &>_{rel} d_j \Leftrightarrow \alpha_{ij} > 0, \\ d_i &<_{rel} d_j \Leftrightarrow \alpha_{ij} < 0. \end{aligned}$$

Thus the estimated gradient is **unbiased w.r.t. document pair preferences**.

Pairwise Differentiable Gradient Descent: Method

Start with initial model θ_t, then indefinitely:

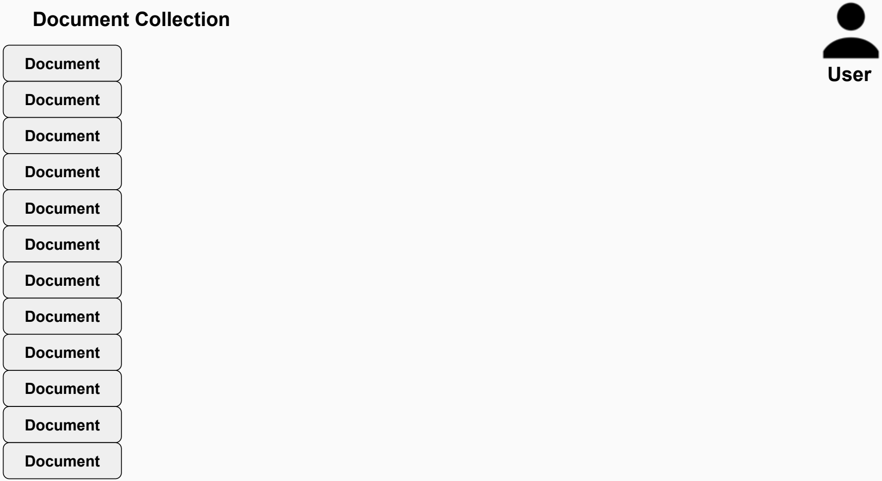
1. Wait for a user query.
2. **Sample** (without replacement) a **ranking** R from the document distribution:

$$P(d|D,\theta_t) = \frac{\exp^{f_{\theta_t}(\mathbf{d})}}{\sum_{d' \in D} \exp^{f_{\theta_t}(\mathbf{d'})}}.$$

3. **Display** the ranking R to the user.
4. **Infer document preferences** from the **user clicks**: $\mathbf{c}$.
5. **Update** model according to the **estimated (unbiased) gradient**:

$$\nabla f_{\theta_t}(\cdot) \approx \sum_{d_i >_{\mathbf{c}} d_j} \rho(d_i, d_j, R) \nabla P(d_i \succ d_j | D, \theta_t).$$

Pairwise Differentiable Gradient Descent: Visualization



Document Collection

Document

Document

Document

Document

Document

Document

Document

Document

Document

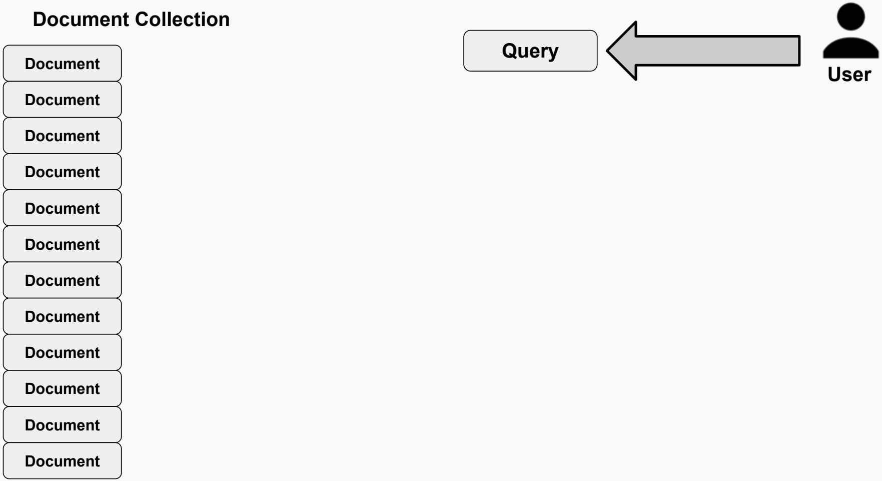
Document

Document

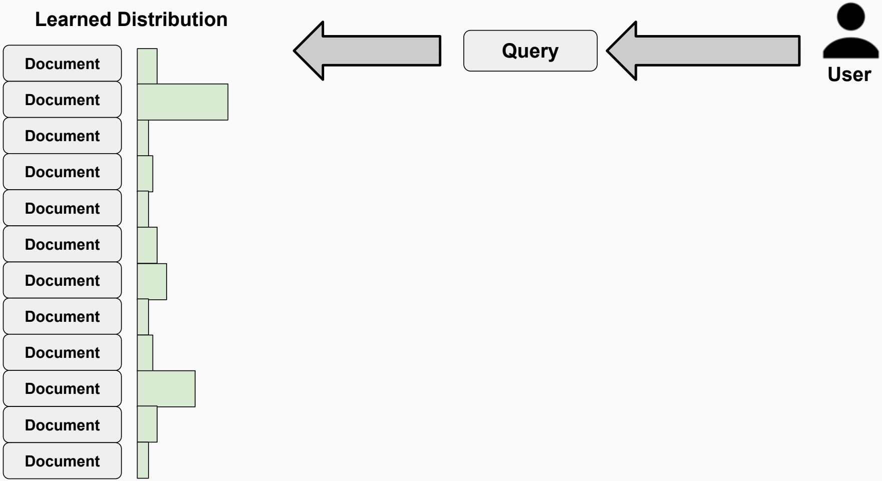
Document

User

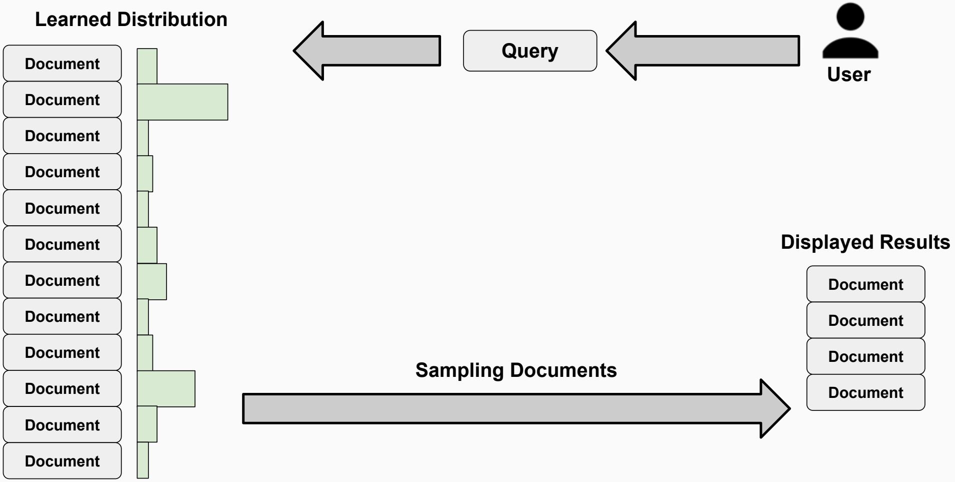
Pairwise Differentiable Gradient Descent: Visualization



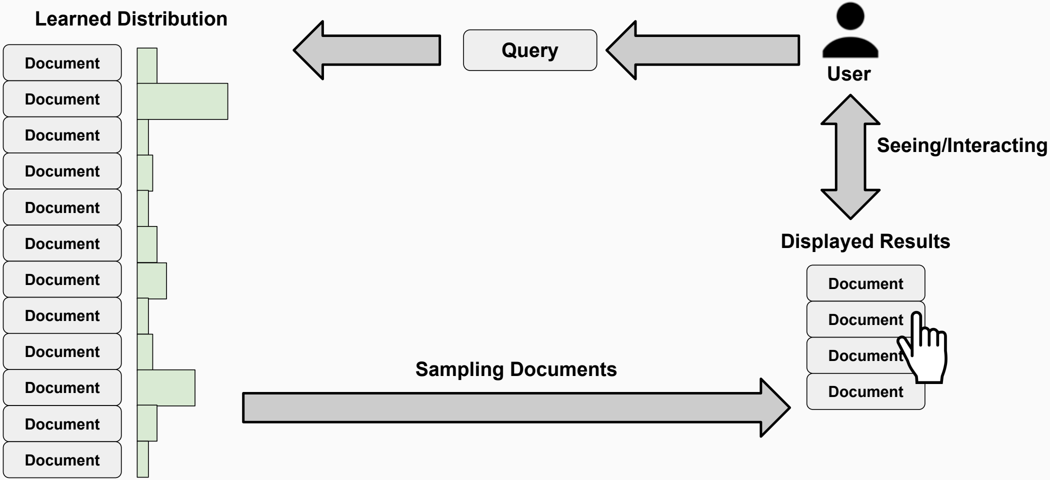
Pairwise Differentiable Gradient Descent: Visualization



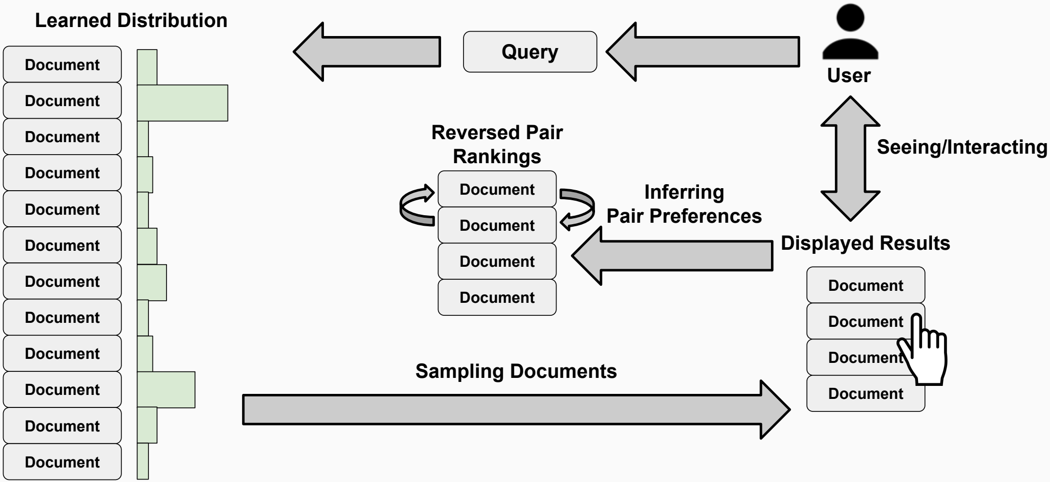
Pairwise Differentiable Gradient Descent: Visualization



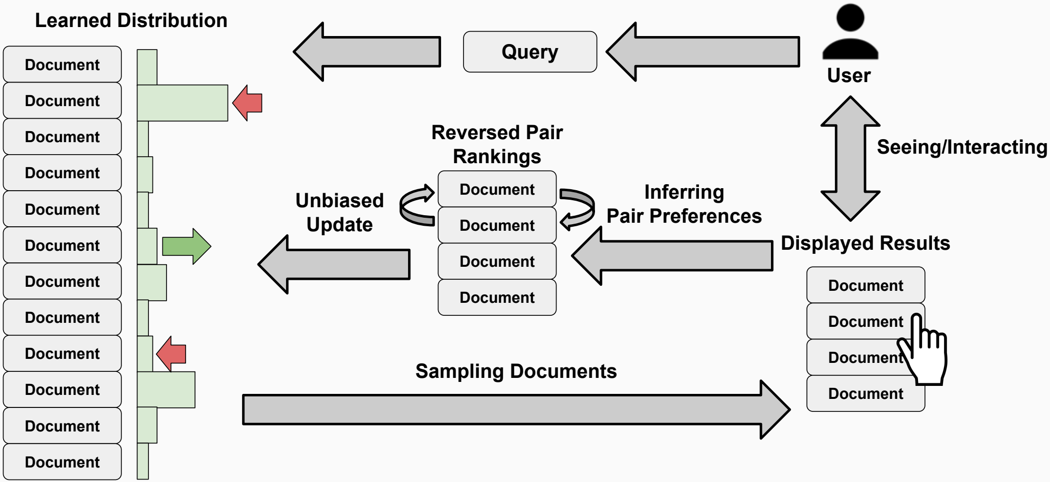
Pairwise Differentiable Gradient Descent: Visualization



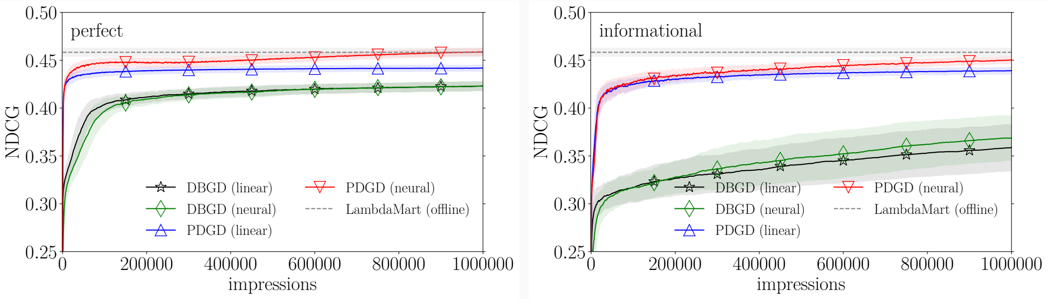
Pairwise Differentiable Gradient Descent: Visualization



Pairwise Differentiable Gradient Descent: Visualization



Pairwise Differentiable Gradient Descent: Results Long Term



Results of simulations on the MSLR-WEB10k dataset, a perfect user (left) and an informational user (right).

Image credits: (Oosterhuis and de Rijke, 2018b).

Comparison of Online Methods

Empirical Comparison: Introduction

Recent most generalized comparison so far (Oosterhuis and de Rijke, 2019).

Simulations based on **largest available industry datasets**:

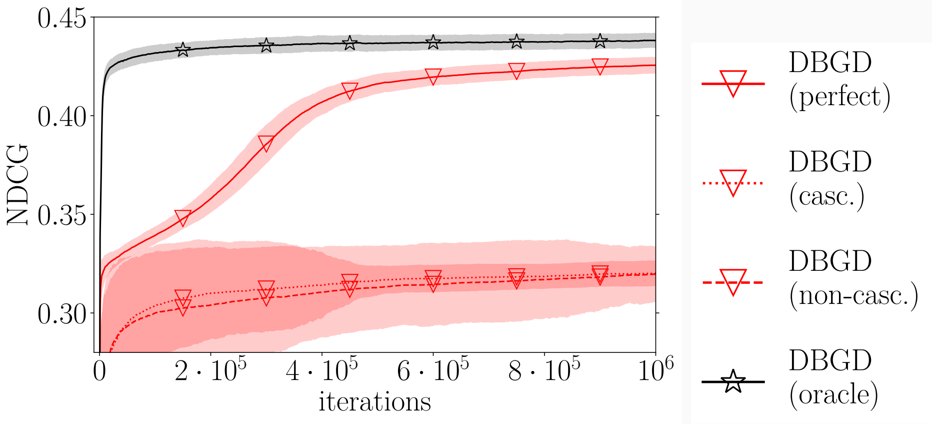
- MSLR-Web10k, Yahoo Webscope, Istella.

Simulated behavior ranging from:

- **ideal:** no noise, no position bias,
- **extremely difficult:** mostly noise, very high position bias.

Dueling Bandit Gradient Descent with an **oracle instead of interleaving**, to see the **maximum potential** of better interleaving methods.

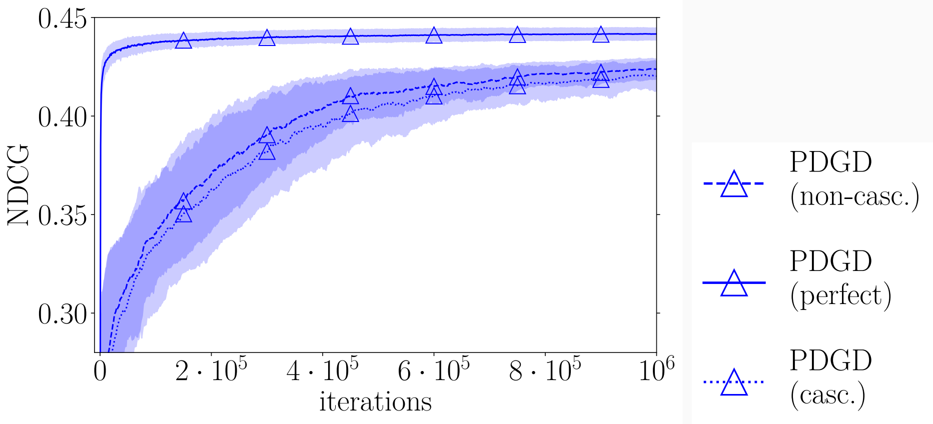
Empirical Comparison: DBGD



Results of simulations on the MSLR-WEB10k dataset.

Image credits: (Oosterhuis and de Rijke, 2019).

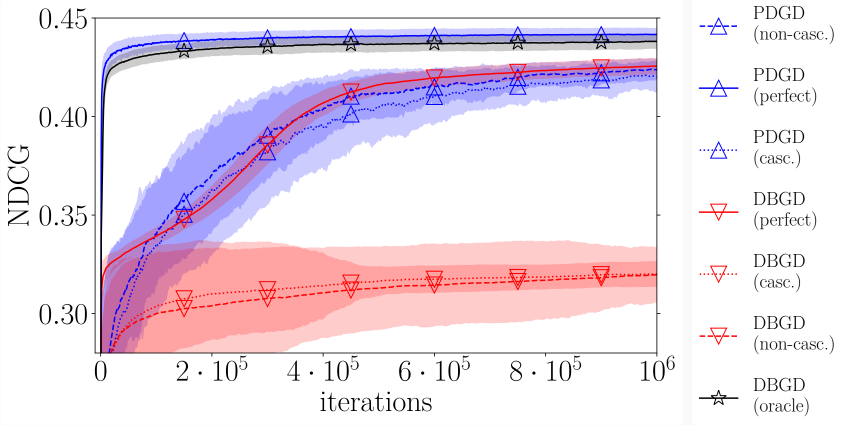
Empirical Comparison: PDGD



Results of simulations on the MSLR-WEB10k dataset.

Image credits: (Oosterhuis and de Rijke, 2019).

Empirical Comparison: All



Results of simulations on the MSLR-WEB10k dataset.

Image credits: (Oosterhuis and de Rijke, 2019).

Empirical Comparison: Conclusion

Dueling Bandit Gradient Descent (DBGD):

- **Unable** to reach **optimal performance** in **ideal** settings.
- Strongly affected by noise and position bias.

Pairwise Differentiable Gradient Descent (PDGD):

- **Capable** of reaching **optimal performance** in ideal settings.
- **Robust** to noise and position bias.
- Considerably **outperforms** DBGD in **all tested experimental settings**.

Theoretical Comparison

Dueling Bandit Based Approaches:

- Sublinear regret bounds proven,
  unsound for ranking problems as commonly applied.
- *Single update steps* are as **unbiased** as its **interleaving method**.

The Differentiable Pairwise Based Approach:

- **No regret bounds** proven.
- *Single update steps* are unbiased w.r.t. **pairwise document preferences**.

For the common ranking problem, neither approach has a theoretical advantage.

The Future for Online Learning to Rank

The **theory** for Online Learning to Rank is **inadequate** and needs **re-evaluation**.

The Dueling Bandit approach appears to be **lacking for optimizing ranking systems**.

Novel alternative approaches have high potential:

- Pairwise Differential Gradient Descent is a clear example.

Comparison of Online LTR with Supervised LTR

Comparison of Online LTR with Supervised LTR

Supervised LTR:

- Uses **manually annotated labels**.
- Optimization is a widely studied and very effective w.r.t. evaluation on annotated labels.
- Often unavailable for practitioners.

Online LTR:

- Learns from **direct interaction**:
    - Debiases by **randomization**.
- Ineffective when applied to historical data.
- Unbiased w.r.t. pairwise preferences.
- Not guaranteed to be unbiased w.r.t. ranking metrics.

Part 4: Conclusion

Part 4: Conclusion

This part will cover the following topics:

- Empirical comparison of methodologies
- Theoretical comparison of methodologies
- Conclusion
- Future directions for unbiased learning to rank

Empirical Comparison of Methodologies

Empirical Comparison

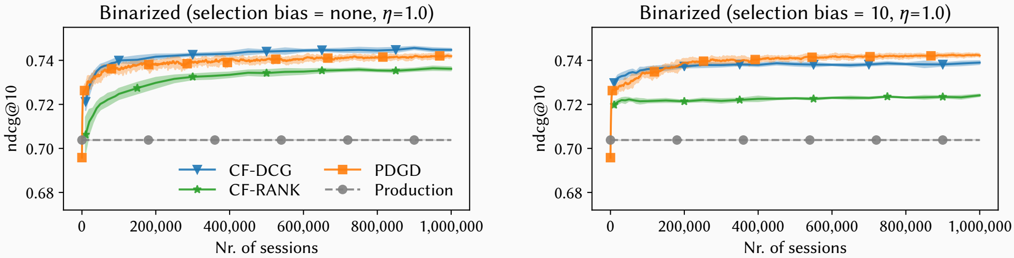
Single empirical comparison so far (Jagerman et al., 2019) was presented at SIGIR'19.

Using the simulated setup common in unbiased learning to rank, we apply both **Inverse Propensity Scoring** and **Pairwise Differentiable Gradient Descent.**

Then we examines the effects of the following factors:

- Number of interactions.
- Degree of **interaction noise**
  (ratio between clicks on relevant and irrelevant documents).
- Degree of **position bias**.
- Presence of **item-selection-bias**, no clicks beyond rank ten.

Empirical Comparison: Item-Selection-Bias

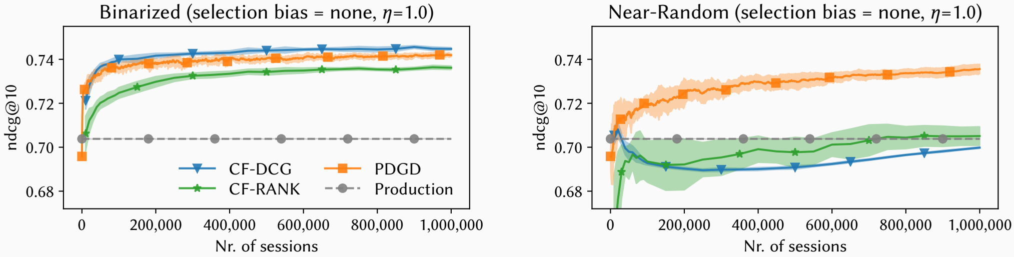


Little interaction noise, no item-selection-bias (left) and at rank ten (right).

The effect of item-selection-bias is greater on the counterfactual method than on the online method.

Image credits: (Jagerman et al., 2019).

Empirical Comparison: Interaction Noise



Little interaction noise (left) and near-random interaction noise (right).

The effect of interaction noise is **substantial** on the counterfactual method and very limited on the online method.

Image credits: (Jagerman et al., 2019).

Empirical Comparison: Conclusion

Counterfactual Learning to Rank:

- Slightly higher performance under:
  - no item-selection-bias,
  - little interaction noise.
- Very affected by high interaction noise.

Online Learning to Rank:

- More reliable performance across settings.
- Handles item-selection bias well.
- More robust to noise

Overall the empirical results suggest that **Online Learning to Rank** is more reliable.

Theoretical Comparison of Methodologies

Theoretical Comparison: Counterfactual Learning to Rank

Counterfactual Learning to Rank:

- Explicitly models position bias.
- Proven to unbiasedly optimize ranking metrics, given that position bias is modelled correctly.
- Can be applied interactively.
- Applicable to any historical interactions.

Theoretical Comparison: Online Learning to Rank

Online Learning to Rank:

- Does not require explicit user model.
- Is not proven to unbiasedly optimize ranking metrics.
- Gradient proven unbiased w.r.t. pairwise document preferences.
- Only effective when applied interactively.
- Not applicable to all historical interactions.

Theoretical Comparison: Conclusion

Counterfactual Learning to Rank:

- Explicit position bias model.
- Proven to unbiasedly optimize ranking metrics.
- Can be interactive.
- Applicable to any historical interactions.

Online Learning to Rank:

- No explicit user model.
- Not proven to unbiasedly optimize ranking metrics.
- Only effective when interactive.
- Not applicable to all historical interactions.

In theory **Counterfactual Learning to Rank** has all the advantageous properties.

Conclusion

Conclusion

- **Supervised approaches** to learning to rank are **limited**.
  - **Annotations often disagree** with user preferences.
- User interactions solve this problem but bring **noise and biases**.
- **Counter-factual approaches** allow for **unbiased** learning to rank:
  - If an **accurate user model** can be learned, we can **adjust for biases**.
  - **Only** uses randomization to **infer a user model**.
- **Online approaches** allow for **unbiased** and **responsive** learning to rank:
  - **Immediately adapt** to user behavior.
  - Perform **randomization** at each step, though limited.
- **Empirically:** Online methods appear to be more reliable.
- **Theoretically:** Counterfactual methods are much more advantageous.

Future Work:
Reinforcement Learning for LTR

Reinforcement Learning

"**Reinforcement learning** is **learning what to do** – how to map situations to actions – so as to maximize a numerical reward signal.
The learner is not told which actions to take, but instead must **discover** which actions yield the most reward **by trying** them.
In the most interesting and challenging cases, actions may affect **not only the immediate reward** but also the next situation and, through that, **all subsequent rewards**."

— Sutton and Barto (1998)

In contrast with supervised learning, both **short and long-term rewards** are considered, and the rewards of actions have to be **discovered by trying** them.

Reinforcement Learning for Ranking

Optimizing for **long-terms** effects seems **very promising**:

- for **long-term user engagement**, across multiple queries/recommendations.
- treat creating a **single ranking** as a **planning** problem:
  - **Diversity**: does placing one document make others less necessary (Xia et al., 2017).
  - **Complex layouts**: what should be displayed and in what form? (Wang et al., 2016b; Oosterhuis and de Rijke, 2018a).
  - **Fairness**: what does a ranking policy do to the marketplace? (Singh and Joachims, 2019).

However, ranking has **difficulties** that make reinforcement learning hard:

- **Massive action space**:
  - all possible rankings over ofter enormous sets of items.
- reinforcement learning requires **large amounts of data**:
  - initial **exploration** phase could **ruin user experience**.

Existing and Future Work

We are not aware of any work that **applies** reinforcement learning to **ranking in practice**.

Work that applies it to **recommendation** does exist.
To avoid the **risky initial exploration** phase, this work focuses on **off-policy learning** (Chen et al., 2019).

This is a **very active area** of research:

- For example, at this conference: (Ma et al., 2020)

Future Directions for Unbiased Learning to Rank

Future Directions

- **The best of both worlds:**
    - The robustness of the online methods.
    - The theoretical properties of the counterfactual methodology.
    - Possibly by using both an explicit user model and randomization during learning.
- **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.

Future Directions

Other areas to expand to:

- **Beyond clicks**:
  - Can we learn from dwell time, conversion, purchases, watch-time, etc.
- **Beyond ten blue links**:
  - Do methods still work in non-traditional displays? (Oosterhuis and de Rijke, 2018a).
  - Are rankings relevant for interactions with virtual assistants?
- **Beyond relevance**:
  - Can we optimize for aspects beside relevance: e.g., result diversity?
  - Or long-term goals: e.g., reinforcement learning (Chen et al., 2019).
- **Responsible A.I.**:
  - Can our algorithms guarantee to respect users during exploration?
  - Can they explain and explicitly substantiate their learned behavior?

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	$\mathit{Prec}@k$	–
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`
- Pairwise Differentiable Gradient Descent and Multileave Gradient Descent: `https://github.com/HarrieO/OnlineLearningToRank`
- 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, X. Wang, C. Li, M. Bendersky, and M. Najork. Addressing trust bias for unbiased learning-to-rank. In *The World Wide Web Conference*, pages 4–14. ACM, 2019b.

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, 2019c.

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.

References ii

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

R. Busa-Fekete and E. Hüllermeier. A survey of preference-based online learning with bandit algorithms. In *International Conference on Algorithmic Learning Theory*, pages 18–39. Springer, 2014.

Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136, 2007.

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

O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems (TOIS)*, 30(1):6, 2012.

M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 456–464. ACM, 2019.

References iii

A. Chuklin, I. Markov, and M. d. Rijke. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7(3):1–115, 2015.

N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*, pages 87–94, 2008.

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.

K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 249–258. ACM, 2011.

K. Hofmann, A. Schuth, S. Whiteson, and M. de Rijke. Reusing historical interaction data for faster online learning to rank for ir. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 183–192. ACM, 2013.

References iv

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. Evaluating retrieval performance using clickthrough data. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining*. Physica Verlag, 2003.

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.

T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *ACM SIGIR Forum*, volume 51, pages 4–11. ACM, 2017a.

References v

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, 2017b.

B. Kveton, C. Szepesvari, Z. Wen, and A. Ashkan. Cascading bandits: Learning to rank in the cascade model. In *International Conference on Machine Learning*, pages 767–776, 2015.

P. Lagrée, C. Vernade, and O. Cappe. Multiple-play bandits in the position-based model. In *Advances in Neural Information Processing Systems*, pages 1597–1605, 2016.

T. Lattimore, B. Kveton, S. Li, and C. Szepesvari. Toprank: A practical algorithm for online stochastic ranking. In *Advances in Neural Information Processing Systems*, pages 3945–3954, 2018.

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.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR '07*, 2007.

References vi

T.-Y. Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

J. Ma, Z. Zhao, X. Yi, J. Yang, M. Chen, J. Tang, L. Hong, and E. H. Chi. Off-policy learning in two-stage recommender systems. In *The World Wide Web Conference*. ACM, 2020.

H. Oosterhuis. Learning to rank and evaluation in the online setting. 12th Russian Summer School in Information Retrieval (RuSSIR 2018), 2018.

H. Oosterhuis and M. de Rijke. Sensitive and scalable online evaluation with theoretical guarantees. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 77–86. ACM, 2017.

H. Oosterhuis and M. de Rijke. Ranking for relevance and display preferences in complex presentation layouts. In *Proceedings of the 41st international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 845–854. ACM, 2018a.

References vii

H. Oosterhuis and M. de Rijke. Differentiable unbiased online learning to rank. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1293–1302. ACM, 2018b.

H. Oosterhuis and M. de Rijke. Optimizing ranking models in an online setting. In *Advances in Information Retrieval*, pages 382–396, Cham, 2019. Springer International Publishing.

H. Oosterhuis, A. Schuth, and M. de Rijke. Probabilistic multileave gradient descent. In *European Conference on Information Retrieval*, pages 661–668. Springer, 2016.

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

F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 245–254. ACM, 2013.

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 43–52. ACM, 2008.

References viii

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.

A. Schuth, H. Oosterhuis, S. Whiteson, and M. de Rijke. Multileave gradient descent for fast online learning to rank. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 457–466. ACM, 2016.

A. Singh and T. Joachims. Policy learning for fairness in ranking. In *Advances in Neural Information Processing Systems*, pages 5427–5437, 2019.

R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

H. Wang, R. Langley, S. Kim, E. McCord-Snook, and H. Wang. Efficient exploration of gradient space for online learning to rank. *arXiv preprint arXiv:1805.07317*, 2018a.

References ix

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, 2016a.

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, 2018b.

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, 2018c.

Y. Wang, D. Yin, L. Jie, P. Wang, M. Yamada, Y. Chang, and Q. Mei. Beyond ranking: Optimizing whole-page presentation. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 103–112, 2016b.

References x

F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, 2008.

L. Xia, J. Xu, Y. Lan, J. Guo, W. Zeng, and X. Cheng. Adapting markov decision process for search result diversification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 535–544, 2017.

Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1201–1208. ACM, 2009.

Y. Yue, Y. Gao, O. Chapelle, Y. Zhang, and T. Joachims. Learning more powerful test statistics for click-based retrieval evaluation. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 507–514. ACM, 2010.

References xi

T. Zhao and I. King. Constructing reliable gradient exploration for online learning to rank. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1643–1652. ACM, 2016.

M. Zoghi, T. Tunys, L. Li, D. Jose, J. Chen, C. M. Chin, and M. de Rijke. Click-based hot fixes for underperforming torso queries. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 195–204, 2016.

Acknowledgments

Ahold Delhaize ICAI NWO vsnu

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