## Learning from user interactions with rankings

*A unification of the field*

Oosterhuis, H.R.

[Link to publication](#)

# Learning from User Interactions with Rankings:

# A Unification of the Field

Harrie Oosterhuis

# Learning from User Interactions with Rankings: A Unification of the Field

**Harrie Oosterhuis**

# Learning from User Interactions with Rankings: A Unification of the Field

<span style="font-variant: small-caps;">Academisch Proefschrift</span>

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op vrijdag 27 november 2020, te 16:00 uur

door

Hendrikus Roelof Oosterhuis

geboren te Schaijk

**Promotiecommissie**

Promotor:

| | |
|---|---|
| Prof. dr. M. de Rijke | Universiteit van Amsterdam |

Co-promotor:

| | |
|---|---|
| Prof. dr. E. Kanoulas | Universiteit van Amsterdam |

Overige leden:

| | |
|---|---|
| Prof. dr. H. Haned | Universiteit van Amsterdam |
| Prof. dr. T. Joachims | Cornell University |
| Dr. ir. J. Kamps | Universiteit van Amsterdam |
| Prof. dr. C.G.M. Snoek | Universiteit van Amsterdam |
| Prof. dr. ir. A.P. de Vries | Radboud Universiteit Nijmegen |

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

# Acknowledgements

Over six years ago, I was invited to join the ILPS research group as an honours MSc. A.I. student. This was the start of an amazing period where I was able to learn, explore and develop myself into a true researcher. Many people have helped me on this journey, and I am truly grateful for all the support and friendship I have received along the way. I hope to inspire future students in the same way that you have all inspired me, and I will now try my best to thank each and everyone of you.

First and foremost, I want to thank Maarten de Rijke, my supervisor and promotor. Maarten, I have learned more from you than I thought possible, you have taught me how to do research, how to become a better teacher and supervisor, and how to develop a research career. Your help was always there when needed and without fail you have always gone above and beyond. You have set a great example for me and all of your students. Thank you so much.

Second, I wish to thank my co-promotor Evangelos Kanoulas. In our annual meetings you have always given me great advice. You are a truly compassionate supervisor, who cares greatly about his students. I am very happy to know that you will continue to have an amazing and caring influence on the future of the research group.

Third, I thank Hinda Haned, Thorsten Joachims, Jaap Kamps, Cees Snoek, and Arjen de Vries, I am truly honoured that you are all part of my PhD committee.

My special thanks to Ana and Tom for being my paranymphs. Through the peaks and valleys of my PhD life you have always been there for me, and I am very honoured to defend this thesis with you on my side.

Another special thanks to Anne Schuth for accepting and supervising me in ILPS when I had only just started the MSc. A.I. In the end, I hold you responsible for my interest in ranking systems and user interactions and I cannot thank you enough. Also I want to thank Petra in particular, your amazing work has made all of this possible. Without contest, I consider you the true ILPS MVP, thank you Petra.

Further thanks to everyone who has been part of ILPS during my journey: Adith, Alexey, Ali, Ali, Amir, Ana, Anna, Anne, Antonis, Arezoo, Arianna, Artem, Bob, Boris, Chang, Christof, Christophe, Chuan, Cristina, Daan, Damien, Dan, Dat, David, David, Dilek, Evgeny, Georgios, Hamid, Hendra, Hinda, Hosein, Ilya, Isaac, Ivan, Jiahuan, Jie, Jin, Julia, Julien, Kaspar, Katya, Ke, Maarten, Maarten, Maartje, Mahsa, Mariya, Marlies, Marzieh, Masrour, Maurits, Mohammad, Mostafa, Mozhdeh, Nikos, Olivier, Peilei, Pengjie, Petra, Praveen, Richard, Ridho, Rolf, Sam, Sami, Sebastian, Shangsong, Shaojie, Spyretta, Svitlana, Thorsten, Tobias, Tom, Trond, Vera, Wanyu, Xiaohui, Xiaojuan, Xinyi, Yangjun, Yaser, Yifan, Zhaochun, and Ziming. Together, you have all made ILPS a wonderful group to be part of, I am very grateful to call all of you my colleagues. I was also very happy to part of several sub-groups that discussed ranking systems, thank you Ali, Arezoo, Artem, Chang, Jin, Julia, Maarten, Rolf, and Wanyu, for the great discussions, hopefully there will be many more discussions to come. In addition, special thanks to Ana, Antonis, Bob, Chang, Hosein, Maartje, Maurits, Nikos, Rolf, and Tom for being great friends as well.

Furthermore, I want to thank the all the people that welcomed me abroad. For the great experiences I had during Google internships I thank Ajay, Ariel, Bo, Eugene,

George, Guan-Lin, Heng-Tze, Larry, Maxime, Michael, Mustafa, Roger, Sujith, Vihan, and Yi-fan. For the absolute amazing time I had in Australia, I want to thank Andrew, Binsheng, Brian, Falk, Joel, Luke, Mark, Sarah, and Shane. Especially Joel and Binsheng for literally travelling to the other side of the world with me. I am really grateful to have met you all and hope the future will allow me to visit you a great many times.

Dan wil ik nog mijn studiegenoten bedanken: Carla, Dasyel, Fabian, Jelle en Wietze, voor de vele leuke herinneringen aan mijn studie. Verder ben ik ook dankbaar voor mijn lange vriendschap met Chiel, Don, Kit, Stefan, Mark en Luuk, het is mij erg dierbaar om vrienden te hebben die ik al sinds de kleuterklas ken.

Als laatste wil ik mijn familie bedanken, de belangrijkste mensen in mijn leven. Marianna, Roelof, Anna en Jeroen, bedankt voor alle steun, zonder jullie was het mij nooit gelukt om zo ver te komen. Ik bedank Helena en Nathalie omdat zij ons altijd zo warm verwelkomen. Erg dankbaar ben ik ook voor mijn lieve oma Anna, die altijd zo geduldig luistert als ik weer eens probeer uit te leggen wat ik nu eigenlijk bij de universiteit doe. Het meest bedank ik mijn grote liefde Emily omdat zij altijd voor mij klaar staat en de dagen zoveel mooier maakt.

<div align="right">

Harrie Oosterhuis
Amsterdam
October 2020

</div>

# Contents

# 1
# Introduction

Search engines allow users to efficiently navigate through the enormous numbers of documents available online [7]. Underlying every search engine is a ranking system that processes documents in order to present a ranking to the user [75]. Over the years, the role of ranking systems has only become more important, as they are now used in a wide variety of settings. Users rely on them to search through many large collections of content, including images [35], scientific articles [58], e-commerce products [60], streaming videos [19], job applications/applicants [32], emails [127], and legal documents [103]. Similarly, ranking systems are used for recommendation as well, where they help to suggest content to users that matches their interests [101]. This may even be content of which users are not aware that they have an interest in [106]. In all these rankings scenarios, the best user experience is provided when the items that users prefer most are on top of the produced rankings [52]. In other words, the ranking should help the user find what they are looking for with the minimal amount of effort [105].

Without a ranking system finding the right information in any sizeable collection becomes an impossible task. Furthermore, without recommendations many online services would lack a lot of user engagement [29]. Thus, ranking systems drive both user satisfaction – providing users with the content they prefer – and user engagement – bringing providers of content or services to interested consumers [34]. Therefore, the performance of a ranking system is very important to both the users of a service and its providers. Due to this importance, a lot of interest has gone to the evaluation of ranking systems [18, 38, 45, 55, 104, 105] and the field of Learning to Rank (LTR) which covers methods for optimizing ranking systems [13, 58–60, 75, 129].

Traditionally, ranking evaluation and LTR methods made use of human judgements in the form of expert annotations [104]: for given pairs of queries and documents, experts are asked to annotate the relevance of a document w.r.t. a specific query. This costly process results in an annotated dataset: a collection of query and document pairs with corresponding expert annotations [17, 27, 76]. For an annotated dataset to be useful it should accurately capture: (i) the queries users typically issue; (ii) the documents that have to be ranked; and (iii) the relevance preferences of the user [105]. With such a dataset, the optimization of a ranking system can be done through supervised LTR methods. These methods optimize ranking metrics such as Precision, Average Relevance Position (ARP) or Discounted Cumulative Gain (DCG), based on the provided relevance annotations [52, 75]. While very important to the LTR field, some severe limitations of

this supervised approach have become apparent over the years: (i) Expert annotations are expensive and time-consuming to obtain [17, 76]. (ii) In sensitive settings acquiring experts annotations can be unethical, for instance, when gathering data for optimizing systems for search over personal documents such as emails [127]. (iii) For specific settings there may be no experts that can judge what is relevant, for instance, in the context of personalized recommendations. (iv) What users perceive as relevant is known to change over time, thus a dataset would have to be updated regularly, further increasing the associated costs [1, 71]. (v) Actual user preferences and expert annotations are often misaligned [104]. Consequently, the supervised approach is infeasible for many LTR practitioners because they do not have the resources to create an annotated dataset or gathering annotations is not possible in their ranking setting. Moreover, even if a dataset can be obtained, it may not lead to the optimal ranking system. Thus there is a need for an alternative to the supervised approach to ranking evaluation and LTR.

An alternative approach that has received a lot of attention is to base evaluation and optimization on user interactions [56, 99]. For rankings this usually means that user clicks are used to compare and improve ranking systems. At first glance user interactions seem to solve the problems with annotations: (i) If a service has enough active users, interactions are virtually free and available at a large scale. (ii) Gathering interactions can be done without showing sensitive items to experts for annotation. (iii) Unlike annotations, interactions are an indication of the actual individual user preferences. Thus there appears to be a lot of potential for using user interactions, however, there are also drawbacks specific to using them: (i) It requires keeping track of large amounts of user behavior, something users may not consent with [94]. (ii) User behavior is very unpredictable, clicks in particular are known to be a very noisy signal [20]. (iii) Clicks are a form of implicit feedback; there are other factors beside user preference that also affect whether a click takes place, making clicks a biased signal of relevance [20, 25]. This thesis will not explore the first drawback and will instead focus on settings where acquiring user interactions is done with consent, in a privacy-respecting and ethical manner. Mainly, we will consider how methods of evaluation and optimization based on clicks can mitigate the negative effects from click-related noise and bias.

Existing methods for ranking evaluation and optimization from user interactions can roughly be divided into two families: the *online* family that deals with bias through direct interaction and result-randomization [100, 132]; and the *counterfactual* family that first models click behavior and then uses the inferred model to correct for bias in logged click data [58, 127]. A further division can be made. For this thesis a decomposition into five areas is relevant. We will divide the online family into three areas:

(i) Online Evaluation – methods like A/B testing and interleaving [56] that interact directly with users to compare ranking systems and randomize displayed results to mitigate biases [18, 44, 110, 111].

(ii) Feature-Based Online LTR – methods like Dueling Bandit Gradient Descent (DBGD) [132] and the Perturbed Preference Perceptron for Ranking (3PR) [100] that optimize feature-based ranking models by direct interaction with users, often relying on online evaluation [42, 111, 126].

(iii) Tabular Online LTR – methods like Cascading Bandits [68] and the Position-Based Model algorithm (PBM) [69] that optimize a single ranking for a single ranking setting, by learning from direct interactions and result randomization [67, 70, 138, 139]. Characteristic about tabular methods is that they do not use any feature-based prediction model but instead memorize the best ranking.

For the counterfactual family, we will use the following division into two areas:

(iv) Counterfactual Evaluation – methods that evaluate rankings based on historically logged clicks. They require an inferred model of click behavior and use that model to correct for biases using, for instance, Inverse Propensity Scoring (IPS) [4, 16, 58, 92, 116].

(v) Counterfactual LTR – methods that use counterfactual evaluation to estimate performance based on historical click-logs, and that optimize ranking models to maximize the estimated a system's performance [2, 3, 46, 58, 92, 127].

This division reveals a rich diversity in approaches that all share the same goal of evaluating or optimizating ranker performance based on user interactions.

On the one hand, this diversity is understandable, since in some settings only one area of methods is applicable. For instance, one cannot add randomization to data that is already logged, making the counterfactual approach the only available option if only logged data is available. On the other hand, the diversity of approaches is also unexpected and raises some questions. For instance, why would online approaches not benefit from an accurate model of click behavior if one is available, similar to the counterfactual approach?

In this thesis, we investigate whether this online/counterfactual division is truly necessary. We will introduce several novel LTR methods that improve over the efficiency of existing methods, and increase the applicability of LTR from user clicks. In particular, we focus on finding LTR methods that bridge the online/counterfactual division and find methods that are highly effective both when applied online or counterfactually. An important result of our thesis on the LTR field, is that we offer a unified perspective and set of LTR methods.

## 1.1   Research Outline and Questions

The overarching question this thesis aims to answer is:

> *Could there be a single general theoretically-grounded approach that has competitive performance for both evaluation and LTR from user clicks on rankings, in both the counterfactual and online settings?*

Our aim is to progress the LTR field towards answering this question in the affirmation. In this thesis, we will explore two directions in search of a single general theoretically-grounded approach. Firstly, by introducing novel online LTR methods that outperform existing online methods in optimization and large scale optimization in the online setting. Secondly, by introducing novel counterfactual LTR methods that build on

the original IPS-based counterfactual LTR approach [58]. Our novel counterfactual LTR methods expand the original counterfactual approach and make it applicable to more tasks and settings. As a result, these novel methods bridge several gaps between counterfactual LTR and the areas of supervised LTR and online LTR. Furthermore, all our novel counterfactual LTR methods are compatible with each other, and can be seen as part of a novel counterfactual LTR framework. At the end of the thesis, our proposed framework has taken the original counterfactual LTR approach and greatly increased its applicability and effectiveness for both online and counterfactual evaluation and optimization. This leads to a more unified perspective of the LTR field, where areas that were previously largely independent are now connected.

### 1.1.1 Novel online methods for learning and evaluating

In the first part of the thesis, we introduce two methods that greatly increase the efficiency of large scale online evaluation and online LTR. Additionally, we take a critical look at several existing methods for online evaluation and online LTR.

Interleaving was introduced as an efficient evaluation paradigm designed for evaluating whether one ranking system outperforms another [56]. Interleaving methods take the rankings produced by two systems and combine them into an interleaved ranking [41, 96, 99]. Clicks on the interleaved ranking are interpreted directly as preference signals between the two systems, resulting in a more data-efficient approach [110]. Thus allowing one to efficiently estimate if an alteration leads to an improved system. Later, the interleaving approach was extended to multileaving which allows for comparisons that include more than two systems at once [12, 108, 109], thereby enabling efficient comparing large numbers of systems with each other.

In Chapter 2 we look at such multileaving methods for large scale online ranking evaluation. Specifically, we investigate the following question:

**RQ1** Does the effectiveness of online ranking evaluation methods scale to large comparisons?

We examine existing multileaving methods in terms of *fidelity* – are they provably unbiased in unambiguous cases [44] – and *considerateness* – are they safe w.r.t. the user experience during the gathering of clicks. From our theoretical analysis, we find that no existing multileaving method manages to meet both criteria. Furthermore, our empirical analysis reveals that their performance decreases as comparisons involve more ranking systems at once. As a novel alternative, we introduce the Pairwise Preference Multileaving (PPM) algorithm, PPM bases evaluation on inferred pairwise item preferences. We prove that it meets both the *fidelity* and *considerateness* criteria. Furthermore, our empirical results indicate that using PPM leads to a much smaller number of errors especially in large scale comparisons.

Besides evaluation, optimization is also very important to obtain effective ranking systems [75]. The idea of optimizing ranking systems based on clicks is long-established. One of the first-theoretically grounded approaches was Dueling Bandit Gradient Descent (DBGD) [132]. For every incoming query, DBGD samples a variation on a ranking system and then uses interleaving to estimate whether this variation is an improvement. If so, it updates the ranking system to be more similar to the variation. Over time

this process is supposed to oscillate towards the optimal ranking system. Numerous extensions have been proposed but all have kept the overall DBGD approach of sampling variations and using online evaluation [42, 100, 111, 126]. This is somewhat puzzling, since this sampling approach is in stark contrast with all other LTR methods that use gradient-based optimization.

In Chapter 3 we explore alternatives to the DBGD approach and ask ourselves the following question:

**RQ2** Is online LTR possible without relying on model-sampling and online evaluation?

We answer this question in the affirmative by proposing a novel online LTR method: Pairwise Differentiable Gradient Descent (PDGD). Unlike DBGD, PDGD does not require model-sampling nor does it make use of any online evaluation. Instead, PDGD optimizes a stochastic Plackett-Luce ranking model and bases its updates on inferred pairwise item preferences. PDGD weights the gradients w.r.t. item-pairs to mitigate the effect of position bias. We prove, under very mild assumptions, that the weighted gradient of PDGD is unbiased w.r.t. item-preferences. Our experimental results show that PDGD requires far fewer interactions to reach the same level of performance as DBGD. Furthermore, we show that even in ideal settings DBGD may not be able to find the optimal model and is ineffective at optimizing neural models. In contrast, PDGD does converge to near optimal models, and reaches even higher performance when applied to neural networks.

The large improvements of PDGD over DBGD observed in Chapter 3, made us wonder whether DBGD is actually a reliable choice for online LTR. In response to this question, Chapter 4 tackles the following question:

**RQ3** Are DBGD LTR methods reliable in terms of theoretical soundness and empirical performance?

First, we take a critical look at the theory underlying the DBGD approach, and find that its assumptions do not hold for deterministic ranking systems and common ranking metrics. Consequently, we conclude that its theory is not applicable to the large majority of existing research that utilizes the DBGD approach [42, 43, 90, 111, 125, 135]. Second, we perform an empirical analysis where DBGD and PDGD are compared in circumstances ranging from near-ideal – where interactions contain little noise and no position bias – to extremely difficult – where interactions contain extreme amounts of noise and position bias. The difference in performance between PDGD and DBGD is so large, that we conclude that PDGD is by far the more reliable choice.

For the field of online LTR this leads us to question the relevancy of DBGD and its extentions, as we have found theoretical weaknesses and empirical inferiority. The fact that virtually all previous methods in the online LTR field are extensions of DBGD raises profound questions.

### 1.1.2 A single framework for online and counterfactual learning to rank

In the second part of the thesis, we expand the existing IPS-based counterfactual LTR approach [58] to create a unified framework for both online and counterfactual LTR and

ranking evaluation based on clicks.

The conclusions of the first part of the thesis revealed that DBGD, which forms the basis of most previous work in online LTR, has problems in terms of performance and its theoretical basis. It is concerning that these conclusions could have been made much earlier: previous work could have taken a critical look at the theory at any moment; furthermore, if previous work had compared DBGD performance with supervised LTR in the prevalent simulated setups, it would have observed the convergence problems of DBGD. To avoid similar issues, we chose to build upon the Counterfactual LTR approach because it has a strong theoretical basis, and additionally, all experimental comparisons in the second part include optimal ranking models to detect potential convergence issues.

In contrast with online LTR approaches, counterfactual LTR and evaluation makes explicit assumptions about user behavior [58, 127]. By making such assumptions, the unbiasedness of counterfactual methods can be proven. Thus guaranteeing optimal convergence, given that the assumptions are correct. While this provides a strong foundation for learning from historically logged clicks, the counterfactual approach is not always applicable nor always the most effective option [50]. The following research questions consider whether counterfactual LTR could overcome its limitations and become the best choice for LTR from clicks in general.

One of the requirements for the unbiasedness of the original counterfactual LTR method is that it requires every relevant item to be displayed at every query [58]. This is a problem in top-$k$ ranking settings where not all items can be displayed at once [92]. Hence, Chapter 5 concerns the question:

**RQ4** Can counterfactual LTR be extended to top-$k$ ranking settings?

We introduce the Policy-Aware estimator that corrects for position bias while taking into account the behavior of a stochastic logging policy. As a result, the policy-aware estimator is unbiased even when learning from top-$k$ feedback, if the policy gives every relevant item a non-zero chance of appearing in the top-$k$. Thus with this extension counterfactual LTR is also applicable to the top-$k$ setting which is especially prevalent in recommendation.

Existing work has considered how to optimize ranking metrics such as DCG using counterfactual LTR [2, 46]. Interestingly, the solutions for counterfactual LTR are very different than those in supervised LTR [13, 129]. To investigate whether this difference is really necessary, Chapter 5 also addresses the question:

**RQ5** Is it possible to apply state-of-the-art supervised LTR methods to the counterfactual LTR problem?

We find that the LambdaLoss framework [129], which includes the famous LambdaMART method [13], can also be applied to counterfactual estimates of ranking metrics. Thus we show that there does not need to be a divide between state-of-the-art supervised LTR and counterfactual LTR.

So far we have not considered the area of tabular online LTR: methods that find the optimal ranking for a single query based on result randomization and direct interaction [67–70, 139]. While these methods need a lot of click data to reach decent

performance, they can always find the optimal ranking since they optimize a memorized ranking, instead of using a feature-based model [138]. The downside is that when few clicks are available for a query, tabular LTR methods are highly sensitive to noise. Thus these approaches are good for specialization: they have great performance on queries where numerous clicks have been observed, while also having an initial period of poor performance. Conversely, counterfactual LTR commonly optimizes feature-based models for generalization to have a robust performance on previously unseen queries, while often not reaching perfect performance at convergence.

Inspired by this contrast, in Chapter 6 we ask ourselves:

**RQ6** Can the specialization ability of tabular online LTR be combined with the robust feature-based approach of counterfactual LTR?

Our answer is in the form of the novel Generalization and Specialization (GENSPEC) algorithm, it optimizes a single robust generalized policy and numerous specialized policies each optimized for a single query. Then the GENSPEC meta-policy uses high-confidence bounds to safely decide per query which policy to deploy. Consequently, for previously unseen queries GENSPEC choose the generalized policy which utilizes the robust feature-based ranking model. While for other queries it can decide to deploy a specialized policy, i.e., if it has enough data to confidently determine that the specialized policy has found the better ranking. For the LTR field, GENSPEC shows that specialization does not need to be unique to tabular online LTR, instead it can be a property of counterfactual LTR as well. Moreover, overall it shows that specialization and generalization are not mutually exclusive abilities.

While counterfactual evaluation methods are designed for using historical clicks, they can be applied online by simply treating newly gathered data as historical [16, 50]. In contrast with online evaluation methods, counterfactual evaluation is completely passive: its methods do not prescribe which rankings should be displayed. This difference leads us to ask the following question in Chapter 7:

**RQ7** Can counterfactual evaluation methods for ranking be extended to perform efficient and effective online evaluation?

We answer this question positively by introducing the novel Logging-Policy Optimization Algorithm (LogOpt) which uses available clicks to optimize the logging policy to minimize the variance of counterfactual estimates of ranking metrics. By minimizing variance, LogOpt increases the data-efficiency of counterfactual evaluation, leading to more accurate estimates from fewer logged clicks. LogOpt is applied when data is still been gathered and changes what rankings will be displayed for future queries. Thus, with the addition of LogOpt, counterfactual evaluation is transformed into an online approach that is actively involved with how data is gathered. Our experimental results suggest that LogOpt is at least as efficient as interleaving methods, while also being proven to be unbiased under the common assumptions of counterfactual LTR.

The results in Chapter 2 and Chapter 7 did not show any online evaluation method converge on a zero error. This lead us to also ask the following question in Chapter 7:

**RQ8** Are existing interleaving methods truly capable of unbiased evaluation w.r.t. position bias?

We prove that, under the assumption of basic position bias, interleaving methods are not unbiased. Furthermore, our results in Chapter 7 indicate that interleaving methods have a systematic error. Unfortunately, we are unable to estimate the impact this systematic error has on real-world comparisons. To the best of our knowledge, no empirical studies have been performed that could measure such a bias, our findings strongly show that such a study would be highly valuable to the field.

In Chapter 7 we have shown that counterfactual ranking evaluation can be as efficient as online evaluation methods, while also having the theoretical justification of counterfactual methods. Naturally this leads to a similar question regarding LTR:

**RQ9** Can the counterfactual LTR approach be extended to perform highly effective online LTR?

In Chapter 8 we answer this question by introducing the intervention-aware estimator for online/counterfactual LTR. The intervention-aware estimator corrects for position-bias and trust-bias while also taking into account the effect of online interventions. This means that if an intervention takes place – i.e., the logging policy changes during the gathering of data – the intervention-aware estimator takes its effect on the interaction biases into account. The result is an estimator that, one the one hand, is just as efficient as other counterfactual estimators when applied to historical data. While on the other hand, it is much more efficient when applied online than existing estimators. Moreover, its performance is comparable to online LTR methods. In contrast with online methods, including DBGD and PDGD, the intervention-aware estimator is proven to be unbiased w.r.t. ranking metrics under the standard assumptions. In other words, it is the only method that is proven to converge on the optimal model, while also being as efficient as the others. Therefore, we consider the intervention-aware estimator a bridge between online and counterfactual LTR as it is a most-reliable choice in both scenarios.

## 1.2  Main Contributions

This section will now summarize the main contributions of this thesis. We differentiate between algorithmic contributions – novel algorithms introduced in the thesis – and theoretical contributions – findings that are important to the field, both in the form of formal proofs and empirical observations.

### 1.2.1  Algorithmic contributions

1. The Pairwise Preference Multileaving (PPM) algorithm for large scale comparisons in online evaluation.

2. The Pairwise Differentiable Gradient Descent (PDGD) algorithm for fast and efficient online LTR.

3. The policy-aware estimator that can perform unbiased counterfactual LTR from top-$k$ settings.

4. Three loss functions for optimizing top-$k$ metrics with counterfactual LTR, including an adaption of the supervised LTR LambdaLoss method.

5. The Generalization and Specialization (GENSPEC) algorithm that combines the specialization ability of tabular models with the generalization ability of feature-based models.

6. The Logging-Policy Optimization Algorithm (LogOpt) algorithm that turns counterfactual evaluation into online evaluation so as to minimize variance by updating the logging policy during the gathering of data.

7. The intervention-aware estimator that bridges the gap between counterfactual and online LTR, by extending the policy-aware estimator to take into account the effect of online interventions.

8. An overarching framework for both online and counterfactual LTR evaluation and optimization, by combining the existing counterfactual approach with the contributions of the second part of the thesis. For counterfactual/online evaluation contributions 3, 6, and 7 can be applied simultaneously, similarly for counterfactual/online LTR the same can be done with contributions 3, 4, 5, and 7.

## 1.2.2 Theoretical contributions

9. An extension of the definition of *fidelity* and *considerateness* for multileaving; in addition, we show that no existing multileave method meets the criteria of both simultaneously.

10. A formal proof that PDGD is unbiased w.r.t. pairwise item preferences under mild assumptions.

11. A formal proof that the assumptions of DBGD are not sound for deterministic ranking models, thus invalidating some claims of unbiasedness in previous online LTR work.

12. An extensive comparison of DBGD and PDGD under circumstances ranging from ideal to near worst-case, revealing that even in ideal circumstances DBGD is often unable to approximate the optimal model.

13. A formal proof for the unbiasedness of the policy-aware and intervention-aware estimators, proving that the former is unbiased w.r.t. position bias and item-selection bias and the latter w.r.t. position bias, item-selection bias, and trust bias respectively.

14. A formal demonstration how LTR loss functions can be adapted to bound top-$k$ metrics, including a description of how LambdaLoss can be adapted for counterfactual LTR.

15. An extension of existing bounds in order to bound the relative performance of two policies, with an additional proof that this bound is more efficient than comparing the bounds of individual policies.

16. A formal proof that interleaving methods are not unbiased w.r.t. position bias.

17. An empirical analysis that reveals that PDGD is not unbiased w.r.t. position bias, item-selection bias, and trust bias, when not applied fully online.

In addition to these contributions, the source code used to perform the experiments in each published chapter has been shared publicly to enable reproducibility.

## 1.3 Thesis Overview

This section will provide an overview of the thesis, and provide some recommendations for reading directions. This thesis consists of an introduction chapter, seven research chapters divided into two parts, and a conclusion. Each research chapter answers one or two of the thesis research questions put forward in Section 1.1, in addition to several chapter-specific research questions. The thesis research questions are important to the overarching story of the thesis, whereas the chapter-specific research questions only consider the individual contributions of the chapters.

The first chapter, which you are currently reading, introduces the subject of this thesis: LTR and ranking evaluation from user clicks. Furthermore, it lays out the thesis research questions this thesis answers, and provides an overview of its contributions and its origins.

Part I titled *Novel Online Methods for Learning and Evaluating* contains three research chapters that all consider online methods for LTR and ranking evaluation. Chapter 2 looks at multileaving methods for online evaluation, evaluates existing methods and introduces a novel multileaving method. Chapter 3 considers online LTR and introduces PDGD, a novel debiased pairwise method. Chapter 4 performs an extensive comparison of the previous state-of-the-art online LTR method DBGD and our novel PDGD, in terms of theoretical guarantees and an experimental analysis.

Part II titled *A Single Framework for Online and Counterfactual Learning to Rank* contains four research chapters that build on the counterfactual approach to LTR and ranker evaluation. The chapters in this part of the thesis are complementary, most of their contributions can be applied together or build upon each other. Chapter 5 extends counterfactual LTR to top-$k$ settings; it introduces a novel estimator to learn from top-$k$ feedback and extends supervised LTR methods to optimize counterfactual estimates of top-$k$ ranking metrics. Chapter 6 looks at both tabular and feature-based ranking models, and introduces an algorithm that optimizes both types of models and safely deploys different models per query. Thus combining the specialization abilities of tabular models with the robust performance of feature-based models in previously unseen circumstances. Chapter 7 aims to unify counterfactual and online ranking evaluation; it introduces a method that updates the logging policy during the gathering of data, turning counterfactual evaluation into efficient online evaluation. Similarly, Chapter 8 seeks to unify counterfactual and online LTR; it proposes a novel estimator that takes into account the effect of online interventions but can also be applied counterfactually. As a result, the estimator is effective for both counterfactual LTR and online LTR.

Lastly, the thesis is concluded in Chapter 9, where we summarize the findings of the thesis; in particular, we discus whether the division between the families of online and counterfactual LTR methods has been bridged. We end the chapter with a discussion of possible future research directions.

The research chapters in this thesis are self-contained, therefore, a reader can read any single chapter independently if they desire. The research chapters grew out of published papers. We wanted to avoid creating alternate versions of published work that deviate from the originals. As a result, the notation between some chapters differ somewhat; to help the reader, we have added a table at the end of each chapter detailing the notation it uses. For the best experience, we recommend reading all the chapters in part II because they build on each other. For the same reason, Chapter 3 and Chapter 4 are best read together.

## 1.4 Origins

We will now list the publications on which the research chapters were based. Each of the publications is a conference paper written by Harrie Oosterhuis and Maarten de Rijke. In all cases, Oosterhuis came up with the main research ideas, performed all experiments, and wrote the majority of text. De Rijke lead the discussions on how each paper should be structured and contributed significantly to the text. In total, this thesis is built on 6 publications [81, 82, 84–88].

**Chapter 2** is based on *Sensitive and scalable online evaluation with theoretical guarantees* published at CIKM '17 by [81].

**Chapter 3** is based on *Differentiable Unbiased Online Learning to Rank* published at CIKM '18 by Oosterhuis and de Rijke [82].

**Chapter 4** is based on *Optimizing Ranking Models in an Online Setting* published at ECIR '19 by Oosterhuis and de Rijke [84].

**Chapter 5** is based on *Policy-Aware Unbiased Learning to Rank for Top-k Rankings* published at SIGIR '20 by Oosterhuis and de Rijke [86].

**Chapter 6** is based on *Robust Generalization and Safe Query-Specialization in Counterfactual Learning to Rank* submitted to WWW '21 by Oosterhuis and de Rijke [87].

**Chapter 7** is based on *Taking the Counterfactual Online: Efficient and Unbiased Online Evaluation for Ranking* published at ICTIR '20 by Oosterhuis and de Rijke [85].

**Chapter 8** is based on *Unifying Online and Counterfactual Learning to Rank* published at WSDM '21 by Oosterhuis and de Rijke [88].

In addition, this thesis also indirectly benefitted from the following publications:

- *Probabilistic Multileave for Online Retrieval Evaluation* published at SIGIR '15 by Schuth et al. [109].

- *Multileave Gradient Descent for Fast Online Learning to Rank* published at WSDM '16 by Schuth et al. [111].

- *Probabilistic Multileave Gradient Descent* published at ECIR '16 by Oosterhuis et al. [90].

- *Balancing Speed and Quality in Online Learning to Rank for Information Retrieval* published at CIKM '17 by Oosterhuis and de Rijke [80].

- *Query-level Ranker Specialization* published at CEUR '17 by Jagerman et al. [49].

- *Ranking for Relevance and Display Preferences in Complex Presentation Layouts* published at SIGIR '18 by Oosterhuis and de Rijke [83].

- *The Potential of Learned Index Structures for Index Compression* published at ADCS '18 by Oosterhuis et al. [91].

- *To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions* published at SIGIR '19 by Jagerman et al. [50].

- *When Inverse Propensity Scoring does not Work: Affine Corrections for Unbiased Learning to Rank* published at CIKM '20 by Vardasbi et al. [123].

- *Keeping Dataset Biases out of the Simulation: A Debiased Simulator for Reinforcement Learning based Recommender Systems* published at RecSys '20 by Huang et al. [47].

Furthermore, other work helped with gaining broader research insights, without being directly related to the thesis topic:

- *Semantic Video Trailers* by Oosterhuis et al. [89].

- *Optimizing Interactive Systems with Data-Driven Objectives* by Li et al. [73].

- *Actionable Interpretability through Optimizable Counterfactual Explanations for Tree Ensembles* by Lucic et al. [77].

# Part I

# Novel Online Methods for Learning and Evaluating

<div style="text-align: right">

# 2

</div>

# Sensitive and Scalable Online Evaluation with Theoretical Guarantees

Multileaved comparison methods generalize interleaved comparison methods to provide a scalable approach for comparing ranking systems based on regular user interactions. Such methods enable the increasingly rapid research and development of search engines. However, existing multileaved comparison methods that provide reliable outcomes do so by degrading the user experience during evaluation. Conversely, current multileaved comparison methods that maintain the user experience cannot guarantee correctness.

In this chapter, we address the following thesis research question:

**RQ1** *Does the effectiveness of online evaluation methods scale to large comparisons?*

Our answer comes in a two-fold contribution; First, we propose a theoretical framework for systematically comparing multileaved comparison methods using the notions of *considerateness*, which concerns maintaining the user experience, and *fidelity*, which concerns reliable correct outcomes. Second, we introduce a novel multileaved comparison method, Pairwise Preference Multileaving (PPM), that performs comparisons based on document-pair preferences, and prove that it is considerate and has fidelity. We show empirically that, compared to previous multileaved comparison methods, PPM is more sensitive to user preferences and scalable with the number of rankers being compared.

## 2.1 Introduction

Evaluation is of tremendous importance to the development of modern search engines. Any proposed change to the system should be verified to ensure it is a true improvement. Online approaches to evaluation aim to measure the actual utility of an Information Retrieval (IR) system in a natural usage environment [45]. Interleaved comparison methods are a within-subject setup for online experimentation in IR. For interleaved comparison, two experimental conditions ("control" and "treatment") are typical. Recently, multileaved comparisons have been introduced for the purpose of efficiently comparing large numbers of rankers [12, 108]. These multileaved comparison methods were introduced as an extension to interleaving and the majority are directly derived

---

This chapter was published as [81]. Appendix 2.A gives a reference for the notation used in this chapter.

from their interleaving counterparts [108, 109]. The effectiveness of these methods has thus far only been measured using simulated experiments on public datasets. While this gives some insight into the general *sensitivity* of a method, there is no work that assesses under what circumstances these methods provide correct outcomes and when they break. Without knowledge of the theoretical properties of multileaved comparison methods we are unable to identify when their outcomes are reliable.

In prior work on interleaved comparison methods a theoretical framework has been introduced that provides explicit requirements that an interleaved comparison method should satisfy [44]. We take this approach as our starting point and adapt and extend it to the setting of multileaved comparison methods. Specifically, the notion of *fidelity* is central to Hofmann et al. [44]'s previous work; Section 2.3 describes the framework with its requirements of *fidelity*. In the setting of multileaved comparison methods, this means that a multileaved comparison method should always recognize an unambiguous winner of a comparison. We also introduce a second notion, *considerateness*, which says that a comparison method should not degrade the user experience, e.g., by allowing all possible permutations of documents to be shown to the user. In this chapter we examine all existing multileaved comparison methods and find that none satisfy both the *considerateness* and *fidelity* requirements. In other words, no existing multileaved comparison method is correct without sacrificing the user experience.

To address this gap, we propose a novel multileaved comparison method, Pairwise Preference Multileaving (PPM). PPM differs from existing multileaved comparison methods as its comparisons are based on inferred pairwise document preferences, whereas existing multileaved comparison methods either use some form of document assignment [108, 109] or click credit functions [12, 108]. We prove that PPM meets both the *considerateness* and the *fidelity* requirements, thus PPM guarantees correct winners in unambiguous cases while maintaining the user experience at all times. Furthermore, we show empirically that PPM is more *sensitive* than existing methods, i.e., it makes fewer errors in the preferences it finds. Finally, unlike other multileaved comparison methods, PPM is computationally efficient and *scalable*, meaning that it maintains most of its *sensitivity* as the number of rankers in a comparison increases.

In this chapter we address thesis research question **RQ1** by answering the following more specific research questions:

**RQ2.1** Does PPM meet the *fidelity* and *considerateness* requirements?

**RQ2.2** Is PPM more sensitive than existing methods when comparing multiple rankers?

To summarize, our contributions in this chapter are:

1. A theoretical framework for comparing multileaved comparison methods;

2. A comparison of all existing multileaved comparison methods in terms of *considerateness*, *fidelity* and *sensitivity*;

3. A novel multileaved comparison method that is *considerate* and has *fidelity* and is more *sensitive* than existing methods.

## 2.2 Related Work

Evaluation of information retrieval systems is a core problem in IR. Two types of approach are common to designing reliable methods for measuring an IR system's effectiveness. Offline approaches such as the Cranfield paradigm [104] are effective for measuring topical relevance, but have difficulty taking into account contextual information including the user's current situation, fast changing information needs, and past interaction history with the system [45]. In contrast, online approaches to evaluation aim to measure the actual utility of an IR system in a natural usage environment. User feedback in online evaluation is usually implicit, in the form of clicks, dwell time, etc.

By far the most common type of controlled experiment on the web is A/B testing [65, 66]. This is a classic between-subject experiment, where each subject is exposed to one of two conditions, *control*—the current system—and *treatment*—an experimental system that is assumed to outperform the control.

An alternative experimental design uses a within-subject setup, where all study participants are exposed to both experimental conditions. Interleaved comparisons [54, 99] have been developed specifically for online experimentation in IR. Interleaved comparison methods have two main ingredients. First, a method for constructing interleaved result lists specifies how to select documents from the original rankings ("control" and "treatment"). Second, a method for inferring comparison outcomes based on observed user interactions with the interleaved result list. Because of their within-subject nature, interleaved comparisons can be up to two orders of magnitude more efficient than A/B tests in effective sample size for studies of comparable dependent variables [18].

For interleaved comparisons, two experimental conditions are typical. Extensions to multiple conditions have been introduced by Schuth et al. [108]. Such *multileaved* comparisons are an efficient online evaluation method for comparing multiple rankers simultaneously. Similar to interleaved comparison methods [41, 56, 96, 99], a multileaved comparison infers preferences between rankers. Interleaved comparisons do this by presenting users with interleaved result lists; these represent two rankers in such a way that a preference between the two can be inferred from clicks on their documents. Similarly, for multileaved comparisons multileaved result lists are created that allow more than two rankers to be represented in the result list. As a consequence, multileaved comparisons can infer preferences between multiple rankers from a single click. Due to this property multileaved comparisons require far fewer interactions than interleaved comparisons to achieve the same accuracy when multiple rankers are involved [108, 109].

The general approach for every multileaved comparison method is described in Algorithm 2.1; here, a comparison of a set of rankers $\mathcal{R}$ is performed over $T$ user interactions. After the user submits a query $q$ to the system (Line 4), a ranking $\mathbf{l}_i$ is generated for each ranker $\mathbf{r}_i$ in $\mathcal{R}$ (Line 6). These rankings are then combined into a single result list by the multileaving method (Line 7); we refer to the resulting list $\mathbf{m}$ as the multileaved result list. In theory a multileaved result list could contain the entire document set, however in practice a length $k$ is chosen beforehand, since users generally only view a restricted number of result pages. This multileaved result list is presented to the user who has the choice to interact with it or not. Any interactions

---

**Algorithm 2.1** General pipeline for multileaved comparisons.

1: **Input**: set of rankers $\mathcal{R}$, documents $D$, no. of timesteps $T$.
2: $P \leftarrow \mathbf{0}$         // *initialize* $|\mathcal{R}| \times |\mathcal{R}|$ *preference matrix*
3: **for** $t = 1, \ldots, T$ **do**
4:     $q_t \leftarrow$ wait_for_user()         // *receive query from user*
5:     **for** $i = 1, \ldots, |\mathcal{R}|$ **do**
6:        $\mathbf{l}_i \leftarrow \mathbf{r}_i(q, D)$         // *create ranking for query per ranker*
7:     $\mathbf{m}_t \leftarrow$ combine_lists$(\mathbf{l}_1, \ldots, \mathbf{l}_R)$         // *combine into multileaved list*
8:     $\mathbf{c} \leftarrow$ display$(\mathbf{m}_t)$         // *display to user and record interactions*
9:     **for** $i = 1, \ldots, |\mathcal{R}|$ **do**
10:        **for** $j = 1, \ldots, |\mathcal{R}|$ **do**
11:           $P_{ij} \leftarrow P_{ij} + $ infer$(i, j, \mathbf{c}, \mathbf{m}_t)$         // *infer pref. between rankers*
12: **return** $P$

---

are recorded in $\mathbf{c}$ and returned to the system (Line 8). While $\mathbf{c}$ could contain any interaction information [63], in practice multileaved comparison methods only consider clicks. Preferences between the rankers in $\mathcal{R}$ can be inferred from the interactions and the preference matrix $P$ is updated accordingly (Line 11). The method of inference (Line 11) is defined by the multileaved comparison method (Line 7). By aggregating the inferred preferences of many interactions a multileaved comparison method can detect preferences of users between the rankers in $\mathcal{R}$. Thus it provides a method of evaluation without requiring a form of explicit annotation.

By instantiating the general pipeline for multileaved comparisons shown in Algorithm 2.1, i.e., the combination method at Line 6 and the inference method at Line 11, we obtain a specific multileaved comparison method. We detail all known multileaved comparison methods in Section 2.4 below.

What we add on top of the work discussed above is a theoretical framework that allows us to assess and compare multileaved comparison methods. In addition, we propose an accurate and scalable multileaved comparison method that is the only one to satisfy the properties specified in our theoretical framework and that also proves to be the most efficient multileaved comparison method in terms of much reduced data requirements.

## 2.3 A Framework for Assessing Multileaved Comparison Methods

Before we introduce a novel multileaved comparison method in Section 2.5, we propose two theoretical requirements for multileaved comparison methods. These theoretical requirements will allow us to assess and compare existing multileaved comparison methods. Specifically, we introduce two theoretical properties: *considerateness* and *fidelity*. These properties guarantee correct outcomes in *unambigious* cases while always maintaining the user experience. In Section 2.4 we show that no currently available multileaved comparison method satisfies both properties. This motivates the introduction of a method that satisfies both properties in Section 2.5.

## 2.3.1 Considerateness

Firstly, one of the most important properties of a multileaved comparison method is how **considerate** it is. Since evaluation is done online it is important that the search experience is not substantially altered [54, 96]. In other words, users should not be obstructed to perform their search tasks during evaluation. As maintaining a user base is at the core of any search engine, methods that potentially degrade the user experience are generally avoided. Therefore, we set the following requirement: the displayed multileaved result list should never show a document $d$ at a rank $i$ if every ranker in $\mathcal{R}$ places it at a lower rank. Writing $r(d, \mathbf{l}_j)$ for the rank of $d$ in the ranking $\mathbf{l}_j$ produced by ranker $\mathbf{r}_j$, this boils down to:

$$\mathbf{m}_i = d \rightarrow \exists \mathbf{r}_j \in \mathcal{R},\ r(d, \mathbf{l}_j) \leq i. \tag{2.1}$$

Requirement 2.1 guarantees that a document can never be displayed higher in a multileaved result list than any ranker would. In addition, it guarantees that if all rankers agree on the top $n$ documents, the resulting multileaved result list $\mathbf{m}$ will display the same top $n$.

## 2.3.2 Fidelity

Secondly, the preferences inferred by a multileaved comparison method should correspond with those of the user with respect to retrieval quality, and should be robust to user behavior that is unrelated to retrieval quality [54]. In other words, the preferences found should be correct in terms of ranker quality. However, in many cases the relative quality of rankers is unclear. For that reason we will use the notion of **fidelity** [44] to compare the correctness of a multileaved comparison method. *Fidelity* was introduced by Hofmann et al. [44] and describes two general cases in which the preference between two rankers is unambiguous. To have *fidelity* the expected outcome of a method is required to be correct in all matching cases. However, the original notion of *fidelity* only considers two rankers as it was introduced for interleaved comparison methods, therefore the definition of *fidelity* must be expanded to the multileaved case. First we describe the following concepts:

**Uncorrelated clicks**  Clicks are considered *uncorrelated* if relevance has no influence on the likelihood that a document is clicked. We write $r(d_i, \mathbf{m})$ for the rank of document $d_i$ in multileaved result list $\mathbf{m}$ and $P(\mathbf{c}_l \mid q, \mathbf{m}_l = d_i)$ for the probability of a click at the rank $l$ at which $d_i$ is displayed: $l = r(d_i, \mathbf{m})$. Then, for a given query $q$

$$uncorrelated(q) \Leftrightarrow \forall l, \forall d_{i,j},\ P(\mathbf{c}_l \mid q, \mathbf{m}_l = d_i) = P(\mathbf{c}_l \mid q, \mathbf{m}_l = d_j). \tag{2.2}$$

**Correlated clicks**  We consider clicks correlated if there is a positive correlation between document relevance and clicks. However we differ from Hofmann et al. [44] by introducing a variable $k$ that denotes at which rank users stop considering documents. Writing $P(\mathbf{c}_i \mid rel(\mathbf{m}_i, q))$ for the probability of a click at rank $i$ if a document relevant

to query $q$ is displayed at this rank, we set

$$
\begin{aligned}
correlated(q, k) \Leftrightarrow \\
\forall i \geq k,\ P(\mathbf{c}_i) = 0 \wedge \forall i < k,\ P(\mathbf{c}_i \mid rel(\mathbf{m}_i, q)) > P(\mathbf{c}_i \mid \neg rel(\mathbf{m}_i, q)).
\end{aligned}
\tag{2.3}
$$

Thus under correlated clicks a relevant document is more likely to be clicked than a non-relevant one at the same rank, if they appear above rank $k$.

**Pareto domination**   Ranker $\mathbf{r}_1$ *Pareto dominates* ranker $\mathbf{r}_2$ if all relevant documents are ranked at least as high by $\mathbf{r}_1$ as by $\mathbf{r}_2$ and $\mathbf{r}_1$ ranks at least one relevant document higher. Writing $rel$ for the set of relevant documents that are ranked above $k$ by at least one ranker, i.e., $rel = \{d \mid rel(d, q) \wedge \exists \mathbf{r}_n \in \mathcal{R}, r(d, \mathbf{l}_n) > k\}$, we require that the following holds for every query $q$ and any rank $k$:

$$
\begin{aligned}
Pareto(\mathbf{r}_i, \mathbf{r}_j, q, k) \Leftrightarrow \\
\forall d \in rel,\ r(d, \mathbf{l}_i) \leq r(d, \mathbf{l}_j) \wedge \exists d \in rel,\ r(d, \mathbf{l}_i) < r(d, \mathbf{l}_j).
\end{aligned}
\tag{2.4}
$$

Then, *fidelity* for multileaved comparison methods is defined by the following two requirements:

1. Under uncorrelated clicks the expected outcome may find no preferences between any two rankers in $\mathcal{R}$:

$$
\forall q, \forall (\mathbf{r}_i, \mathbf{r}_j) \in \mathcal{R}, \quad uncorrelated(q) \Rightarrow E[P_{ij} \mid q] = 0.
\tag{2.5}
$$

2. Under correlated clicks, a ranker that Pareto dominates all other rankers must win the multileaved comparison in expectation:

$$
\begin{aligned}
\forall k, \forall q, \forall \mathbf{r}_i \in \mathcal{R}, \\
\big(correlated(q, k) \wedge \forall \mathbf{r}_j \in \mathcal{R},\ i \neq j \to Pareto(\mathbf{r}_i, \mathbf{r}_j, q, k)\big) \\
\Rightarrow (\forall \mathbf{r}_j \in \mathcal{R}, i \neq j \to E[P_{ij} \mid q] > 0).
\end{aligned}
\tag{2.6}
$$

Note that for the case where $|\mathcal{R}| = 2$ and if only $k = |D|$ is considered, these requirements are the same as for interleaved comparison methods [44]. The $k$ parameter was added to allow for *fidelity* in *considerate* methods, since it is impossible to detect preferences at ranks that users never consider without breaking the *considerateness* requirement. We argue that differences at ranks that users are not expected to observe should not affect comparison outcomes. *Fidelity* is important for a multileaved comparison method as it ensures that an unambiguous winner is expected to be identified. Additionally, the first requirement ensures that in exception no preferences are inferred when clicks are unaffected by relevancy.

## 2.3.3   Additional properties

In addition to the two theoretical properties listed above, considerateness and fidelity, we also scrutinize multileaved comparison methods to determine whether they accurately find preferences between all rankers in $\mathcal{R}$ and minimize the number of user impressions required do so. This empirical property is commonly known as **sensitivity** [44, 108].

In Section 2.6 we describe experiments that are aimed at comparing the sensitivity of multileaved comparison methods. Here, two aspects of every comparison are considered: the level of error at which a method converges and the number of impressions required to reach that level. Thus, an interleaved comparison method that learns faster initially but does not reach the same final level of error is deemed worse.

## 2.4 An Assessment of Existing Multileaved Comparison Methods

We briefly examine all existing multileaved comparison methods to determine whether they meet the *considerateness* and *fidelity* requirements. An investigation of the empirical sensitivity requirement is postponed until Section 2.6 and 2.7.

### 2.4.1 Team Draft Multileaving

Team-Draft Multileaving (TDM) was introduced by Schuth et al. [108] and is based on the previously proposed Team Draft Interleaving (TDI) [99]. Both methods are inspired by how team assignments are often chosen for friendly sport matches. The multileaved result list is created by sequentially sampling rankers without replacement; the first sampled ranker places their top document at the first position of the multileaved list. Subsequently, the next sampled ranker adds their top pick of the remaining documents. When all rankers have been sampled, the process is continued by sampling from the entire set of rankers again. The method is stops when all documents have been added. When a document is clicked, TDM assigns the click to the ranker that contributed the document. For each impression binary preferences are inferred by comparing the number of clicks each ranker received.

It is clear that TDM is *considerate* since each added document is the top pick of at least one ranker. However, TDM does not meet the fidelity requirements. This is unsurprising as previous work has proven that TDI does not meet these requirements [41, 44, 96]. Since TDI is identical to TDM when the number of rankers is $|\mathcal{R}| = 2$, TDM does not have *fidelity* either.

### 2.4.2 Optimized Multileaving

Optimized Multileaving (OM) was proposed by Schuth et al. [108] and serves as an extension of Optimized Interleaving (OI) introduced by Radlinski and Craswell [96]. The allowed multileaved result lists of OM are created by sampling rankers with replacement at each iteration and adding the top document of the sampled ranker. However, the probability that a multileaved result list is shown is not determined by the generative process. Instead, for a chosen credit function OM performs an optimization that computes a probability for each multileaved result list so that the expected outcome is unbiased and sensitive to correct preferences.

All of the allowed multileaved result lists of OM meet the *considerateness* requirement, and in theory instantiations of OM could have *fidelity*. However, in practice OM does not meet the *fidelity* requirements. There are two main reasons for this. First, it

is not guaranteed that a solution exists for the optimization that OM performs. For the interleaving case this was proven empirically when $k = 10$ [96]. However, this approach does not scale to any number of rankers. Secondly, unlike OI, OM allows more result lists than can be computed in a feasible amount of time. Consider the top $k$ of all possible multileaved result lists; in the worst case this produces $|\mathcal{R}|^k$ lists. Computing all lists for a large value of $|\mathcal{R}|$ and performing linear constraint optimization over them is simply not feasible. As a solution, Schuth et al. [108] propose a method that samples from the allowed multileaved result lists and relaxes constraints when there is no exact solution. Consequently, there is no guarantee that this method does not introduce bias. Together, these two reasons show that the *fidelity* of OI does not imply fidelity of OM. It also shows that OM is computationally very costly.

### 2.4.3 Probabilistic Multileaving

Probabilistic Multileaving (PM) [109] is an extension of Probabilistic Interleaving (PI) [41], which was designed to solve the flaws of TDI. Unlike the previous methods, PM considers every ranker as a distribution over documents, which is created by applying a soft-max to each of them. A multileaved result list is created by sampling a ranker with replacement at each iteration and sampling a document from the ranker that was selected. After the sampled document has been added, all rankers are renormalized to account for the removed document. During inference PM credits every ranker the expected number of clicked documents that were assigned to them. This is done by marginalizing over the possible ways the list could have been constructed by PM. A benefit of this approach is that it allows for comparisons on historical data [41, 44].

A big disadvantage of PM is that it allows any possible ranking to be shown, albeit not with uniform probabilities. This is a big deterrent for the usage of PM in operational settings. Furthermore, it also means that PM does not meet the *considerateness* requirement. On the other hand, PM does meet the *fidelity* requirements, the proof for this follows from the fact that every ranker is equally likely to add a document at each location in the ranking. Moreover, if multiple rankers want to place the same document somewhere they have to share the resulting credits.[1] Similar to OM, PM becomes infeasible to compute for a large number of rankers $|\mathcal{R}|$; the number of assignments in the worst case is $|R|^k$. Fortunately, PM inference can be estimated by sampling assignments in a way that maintains *fidelity* [90, 109].

### 2.4.4 Sample Only Scored Multileaving

Sample-Scored-Only Multileaving (SOSM) was introduced by Brost et al. [12] in an attempt to create a more scalable multileaved comparison method. It is the only existing multileaved comparison method that does not have an interleaved comparison counterpart. SOSM attempts to increase *sensitivity* by ignoring all non-sampled documents during inference. Thus, at each impression a ranker receives credits according to how it ranks the documents that were sampled for the displayed multileaved result list of size $k$. The preferences at each impression are made binary before being added to the mean.

---

[1]Brost et al. [12] proved that if the preferences at each impression are made binary the *fidelity* of PM is lost.

Table 2.1: Overview of multileaved comparison methods and whether they meet the *considerateness* and *fidelity* requirements.

| | **Considerateness** | **Fidelity** | **Source** |
|---|---|---|---|
| TDM | ✓ | | [108] |
| OM | ✓ | | [108] |
| PM | | ✓ | [109] |
| SOSM | ✓ | | [12] |
| PPM | ✓ | ✓ | this chapter |

SOSM creates multileaved result lists following the same procedure as TDM, a choice that seems arbitrary.

SOSM meets the *considerateness* requirements for the same reason TDM does. However, SOSM does not meet the fidelity requirement. We can prove this by providing an example where preferences are found under uncorrelated clicks. Consider the two documents $A$ and $B$ and the three rankers with the following three rankings:

$$\mathbf{l}_1 = AB, \qquad \mathbf{l}_2 = \mathbf{l}_3 = BA.$$

The first requirement of fidelity states that under uncorrelated clicks no preferences may be found in expectation. Uncorrelated clicks are unconditioned on document relevance (Equation 2.2); however, it is possible that they display position bias [134]. Thus the probability of a click at the first rank may be greater than at the second:

$$P(\mathbf{c}_1 \mid q) > P(\mathbf{c}_2 \mid q).$$

Under position biased clicks the expected outcome for each possible multileaved result list is not zero. For instance, the following preferences are expected:

$$E[P_{12} \mid \mathbf{m} = AB] > 0,$$
$$E[P_{12} \mid \mathbf{m} = BA] < 0,$$
$$E[P_{12} \mid \mathbf{m} = AB] = -E[P_{12} \mid \mathbf{m} = BA].$$

Since SOSM creates multileaved result lists following the TDM procedure the probability $P(\mathbf{m} = BA)$ is twice as high as $P(\mathbf{m} = AB)$. As a consequence, the expected preference is biased against the first ranker:

$$E[P_{12}] < 0.$$

Hence, SOSM does not have *fidelity*. This outcome seems to stem from a disconnect between how multileaved results lists are created and how preferences are inferred.

To conclude this section, Table 2.1 provides an overview of our findings thus far, i.e., the theoretical requirements that each multileaved comparison method satisfies; we have also included PPM, the multileaved comparison method that we will introduce below.

---

**Algorithm 2.2** Multileaved result list construction for PPM.

---

1: **Input**: set of rankers $\mathcal{R}$, rankings $\{\mathbf{l}\}$, documents $D$.
2: $\mathbf{m} \leftarrow []$                        *// initialize empty multileaving*
3: **for** $n = 1, \ldots, |D|$ **do**
4:     $\hat{\Omega}_n \leftarrow \Omega(n, \mathcal{R}, D) \setminus \mathbf{m}$           *// choice set of remaining documents*
5:     $d \leftarrow \text{uniform\_sample}(\hat{\Omega}_n)$          *// uniformly sample next document*
6:     $\mathbf{m} \leftarrow \text{append}(\mathbf{m}, d)$          *// add sampled document to multileaving*
7: **return** $\mathbf{m}$

---

## 2.5   A Novel Multileaved Comparison Method

The previously described multileaved comparison methods are based around direct credit assignment, i.e., credit functions are based on single documents. In contrast, we introduce a method that estimates differences based on pairwise document preferences. We prove that this novel method is the only multileaved comparison method that meets the *considerateness* and *fidelity* requirements set out in Section 2.3.

The multileaved comparison method that we introduce is Pairwise Preference Multileaving (PPM). It infers pairwise preferences between documents from clicks and bases comparisons on the agreement of rankers with the inferred preferences. PPM is based on the assumption that a clicked document is preferred to: (i) all of the unclicked documents above it; and (ii) the next unclicked document. These assumptions are long-established [55] and form the basis of pairwise Learning to Rank (LTR) [54].

We write $\mathbf{c}_{r(d_i, \mathbf{m})}$ for a click on document $d_i$ displayed in multileaved result list $\mathbf{m}$ at the rank $r(d_i, \mathbf{m})$. For a document pair $(d_i, d_j)$, a click $\mathbf{c}_{r(d_i, \mathbf{m})}$ infers a preference as follows:

$$c_{r(d_i, \mathbf{m})} \wedge \neg c_{r(d_j, \mathbf{m})} \wedge \left( \exists i, (c_i \wedge r(d_j, \mathbf{m}) < i) \vee c_{r(d_j, \mathbf{m}) - 1} \right) \Leftrightarrow d_i >_c d_j. \quad (2.7)$$

In addition, the preference of a ranker $\mathbf{r}$ is denoted by $d_i >_\mathbf{r} d_j$. Pairwise preferences also form the basis for Preference-Based Balanced Interleaving (PBI) introduced by He et al. [38]. However, previous work has shown that PBI does not meet the *fidelity* requirements [44]. Therefore, we do not use PBI as a starting point for PPM. Instead, PPM is derived directly from the *considerateness* and *fidelity* requirements. Consequently, PPM constructs multileaved result lists inherently differently and its inference method has *fidelity*, in contrast with PBI.

When constructing a multileaved result list $\mathbf{m}$ we want to be able to infer unbiased preferences while simultaneously being *considerate*. Thus, with the requirement for *considerateness* in mind we define a choice set as:

$$\Omega(i, \mathcal{R}, D) = \{d \mid d \in D \wedge \exists \mathbf{r}_j \in \mathcal{R}, r(d, \mathbf{l}_j) \leq i\}. \quad (2.8)$$

This definition is chosen so that any document in $\Omega(i, \mathcal{R}, D)$ can be placed at rank $i$ without breaking the *considerateness* requirement (Equation 2.1). The multileaving method of PPM is described in Algorithm 2.2. The approach is straightforward: at each rank $n$ the set of documents $\hat{\Omega}_n$ is determined (Line 4). This set of documents is $\Omega(n, \mathcal{R}, D)$ with the previously added documents removed to avoid document repetition.

---

---

**Algorithm 2.3** Preference inference for PPM.

1: **Input**: rankers $\mathcal{R}$, rankings $\{\mathbf{l}\}$, documents $D$, multileaved result list $\mathbf{m}$, clicks $\mathbf{c}$.
2: $P \leftarrow \mathbf{0}$                                                           *// preference matrix of $|\mathcal{R}| \times |\mathcal{R}|$*
3: **for** $(d_i, d_j) \in \{(d_i, d_j) \mid d_i >_\mathbf{c} d_j\}$ **do**
4:    **if** $\underline{r}(i, j, \mathbf{m}) \geq \bar{r}(i, j)$ **then**
5:        $w \leftarrow 1$                                     *// variable to store $P(\underline{r}(i, j, \mathbf{m}) \geq \bar{r}(i, j))$*
6:        $min\_x \leftarrow \min_{d \in \{d_i, d_j\}} \min_{\mathbf{r}_n \in \mathcal{R}} r(d, \mathbf{l}_n)$
7:        **for** $x = min\_x, \ldots, \bar{r}(i, j) - 1$ **do**
8:            $w \leftarrow w \cdot (1 - (|\Omega(x, \mathcal{R}, D)| - x - 1)^{-1})$
9:        **for** $n = 1, \ldots, |R|$ **do**
10:           **for** $m = 1, \ldots, |R|$ **do**
11:               **if** $d_i >_{\mathbf{r}_n} d_j \wedge n \neq m$ **then**
12:                   $P_{nm} \leftarrow P_{nm} + w^{-1}$                      *// result of scoring function $\phi$*
13:               **else if** $n \neq m$ **then**
14:                   $P_{nm} \leftarrow P_{nm} - w^{-1}$
15: **return** $P$

---

Then, the next document is sampled uniformly from $\hat{\Omega}_n$ (Line 5), thus every document in $\hat{\Omega}_n$ has a probability:

$$\frac{1}{|\Omega(n, \mathcal{R}, D)| - n + 1} \tag{2.9}$$

of being placed at position $n$ (Line 6). Since $\hat{\Omega}_n \subseteq \Omega(n, \mathcal{R}, D)$ the resulting $\mathbf{m}$ is guaranteed to be *considerate*.

While the multileaved result list creation method used by PPM is simple, its preference inference method is more complicated as it has to meet the *fidelity* requirements. First, the preference found between a ranker $\mathbf{r}_n$ and $\mathbf{r}_m$ from a single interaction $\mathbf{c}$ is determined by:

$$P_{nm} = \sum_{d_i >_\mathbf{c} d_j} \phi(d_i, d_j, \mathbf{r}_n, \mathbf{m}, \mathcal{R}) - \phi(d_i, d_j, \mathbf{r}_m, \mathbf{m}, \mathcal{R}), \tag{2.10}$$

which sums over all document pairs $(d_i, d_j)$ where interaction $\mathbf{c}$ inferred a preference. Before the scoring function $\phi$ can be defined we introduce the following function:

$$\bar{r}(i, j, \mathcal{R}) = \max_{d \in \{d_i, d_j\}} \min_{\mathbf{r}_n \in \mathcal{R}} r(d, \mathbf{l}_n). \tag{2.11}$$

For succinctness we will note $\bar{r}(i, j) = \bar{r}(i, j, \mathcal{R})$. Here, $\bar{r}(i, j)$ provides the highest rank at which both documents $d_i$ and $d_j$ can appear in $\mathbf{m}$. Position $\bar{r}(i, j)$ is important to the document pair $(d_i, d_j)$, since if both documents are in the remaining documents $\hat{\Omega}_{\bar{r}(i,j)}$, then the rest of the multileaved result list creation process is identical for both. To keep notation short we introduce:

$$\underline{r}(i, j, \mathbf{m}) = \min_{d \in \{d_i, d_j\}} r(d, \mathbf{m}). \tag{2.12}$$

---

Therefore, if $\underline{r}(i, j, \mathbf{m}) \geq \bar{r}(i, j)$ then both documents appear below $\bar{r}(i, j)$. This, in turn, means that both documents are equally likely to appear at any rank:

$$\forall n, \, P(\mathbf{m}_n = d_i \mid \underline{r}(i, j, \mathbf{m}) \geq \bar{r}(i, j))$$
$$= P(\mathbf{m}_n = d_j \mid \underline{r}(i, j, \mathbf{m}) \geq \bar{r}(i, j)). \quad (2.13)$$

The scoring function $\phi$ is then defined as follows:

$$\phi(d_i, d_j, \mathbf{r}, \mathbf{m}) = \begin{cases} 0, & \underline{r}(i, j, \mathbf{m}) < \bar{r}(i, j) \\ \frac{-1}{P(\underline{r}(i,j,\mathbf{m}) \geq \bar{r}(i,j))}, & d_i <_{\mathbf{r}} d_j \\ \frac{1}{P(\underline{r}(i,j,\mathbf{m}) \geq \bar{r}(i,j))}, & d_i >_{\mathbf{r}} d_j, \end{cases} \quad (2.14)$$

indicating that a zero score is given if one of the documents appears above $\bar{r}(i, j)$. Otherwise, the value of $\phi$ is positive or negative depending on whether the ranker $\mathbf{r}$ agrees with the inferred preference between $d_i$ and $d_j$. Furthermore, this score is inversely weighted by the probability $P(\underline{r}(i, j, \mathbf{m}) \geq \bar{r}(i, j))$. Therefore, pairs that are less likely to appear below their threshold $\bar{r}(i, j)$ result in a higher score than for more commonly occuring pairs. Algorithm 2.3 displays how the inference of PPM can be computed. The scoring function $\phi$ was carefully chosen to guarantee *fidelity*, the remainder of this section will sketch the proof for PPM meeting its requirements.

The two requirements for *fidelity* will be discussed in order:

### Requirement 1

The first fidelity requirement states that under uncorrelated clicks the expected outcome should be zero. Consider the expected preference:

$$E[P_{nm}] = \sum_{d_i, d_j} \sum_{\mathbf{m}} P(d_i >_c d_j \mid \mathbf{m}) P(\mathbf{m}) (\phi(d_i, d_j, \mathbf{r}_n, \mathbf{m})$$
$$- \phi(d_i, d_j, \mathbf{r}_m, \mathbf{m})). \quad (2.15)$$

To see that $E[P_{nm}] = 0$ under uncorrelated clicks, take any multileaving $\mathbf{m}$ where $P(\mathbf{m}) > 0$ and $\phi(d_i, d_j, \mathbf{r}, \mathbf{m}) \neq 0$ with $\mathbf{m}_x = d_i$ and $\mathbf{m}_y = d_j$. Then there is always a multileaved result list $\mathbf{m}'$ that is identical expect for swapping the two documents so that $\mathbf{m}'_x = d_j$ and $\mathbf{m}'_y = d_i$. The scoring function only gives non-zero values if both documents appear below the threshold $\underline{r}(i, j, \mathbf{m}) < \bar{r}(i, j)$ (Equation 2.14). At this point the probability of each document appearing at any position is the same (Equation 2.13), thus the following holds:

$$P(\mathbf{m}) = P(\mathbf{m}'), \quad (2.16)$$
$$\phi(d_i, d_j, \mathbf{r}_n, \mathbf{m}) = -\phi(d_j, d_i, \mathbf{r}_n, \mathbf{m}'). \quad (2.17)$$

Finally, from the definition of uncorrelated clicks (Equation 2.2) the following holds:

$$P(d_i >_c d_j \mid \mathbf{m}) = P(d_j >_c d_i \mid \mathbf{m}'). \quad (2.18)$$

As a result, any document pair $(d_i, d_j)$ and multileaving $\mathbf{m}$ that affects the expected outcome is cancelled by the multileaving $\mathbf{m}'$. Therefore, we can conclude that $E[P_{nm}] = 0$ under uncorrelated clicks, and that PPM meets the first requirement of *fidelity*.

### Requirement 2

The second *fidelity* requirement states that under correlated clicks a ranker that Pareto dominates all other rankers should win the multileaved comparison. Therefore, the expected value for a Pareto dominating ranker $r_n$ should be:

$$\forall m, n \neq m \rightarrow E[P_{nm}] > 0. \tag{2.19}$$

Take any other ranker $\mathbf{r}_m$ that is thus Pareto dominated by $\mathbf{r}_n$. The proof for the first requirement shows that $E[P_{nm}]$ is not affected by any pair of documents $d_i, d_j$ with the same relevance label. Furthermore, any pair on which $\mathbf{r}_n$ and $\mathbf{r}_m$ agree will not affect the expected outcome since:

$$(d_i >_{\mathbf{r}_n} d_j \leftrightarrow d_i >_{\mathbf{r}_m} d_j) \Rightarrow \phi(d_i, d_j, \mathbf{r}_n, \mathbf{m}) - \phi(d_i, d_j, \mathbf{r}_m, \mathbf{m}) = 0. \tag{2.20}$$

Then, for any relevant document $d_i$, consider the set of documents that $\mathbf{r}_n$ incorrectly prefers over $d_i$:

$$A = \{d_j \mid \neg rel(d_j) \wedge d_j >_{\mathbf{r}_n} d_i\} \tag{2.21}$$

and the set of documents that $\mathbf{r}_m$ incorrectly prefers over $d_i$ and places higher than where $\mathbf{r}_n$ places $d_i$:

$$B = \{d_j \mid \neg rel(d_j) \wedge d_j >_{\mathbf{r}_m} d_i \wedge r(d_j, \mathbf{l}_m) < r(d_i, \mathbf{l}_n)\}. \tag{2.22}$$

Since $\mathbf{r}_n$ Pareto dominates $\mathbf{r}_m$, it has the same or fewer incorrect preferences: $|A| \leq |B|$. Furthermore, for any document $d_j$ in either $A$ or $B$ the threshold of the pair $d_i, d_j$ is the same:

$$\forall d_j \in A \cup B, \bar{r}(i, j) = r(d_i, \mathbf{l}_n). \tag{2.23}$$

Therefore, all pairs with documents from $A$ and $B$ will only get a non-zero value from $\phi$ if they both appear at or below $r(d_i, \mathbf{l}_n)$. Then, using Equation 2.13 and the Bayes rule we see:

$$\forall (d_j, d_l) \in A \cup B, \quad \frac{P(\mathbf{m}_x = d_j, \underline{r}(i, j, \mathbf{m}) \geq \bar{r}(i, j, \mathcal{R}))}{P(\underline{r}(i, j, \mathbf{m}) \geq \bar{r}(i, j, \mathcal{R}))} \\ = \frac{P(\mathbf{m}_x = d_l, \underline{r}(i, l, \mathbf{m}) \geq \bar{r}(i, l, \mathcal{R}))}{P(\underline{r}(i, l, \mathbf{m}) \geq \bar{r}(i, l, \mathcal{R}))}. \tag{2.24}$$

Similarly, the reweighing of $\phi$ ensures that every pair in $A$ and $B$ contributes the same to the expected outcome. Thus, if both rankers rank $d_i$ at the same position the following sum:

$$\sum_{d_j \in A \cup B} \sum_{\mathbf{m}} P(\mathbf{m})$$

$$\cdot [P(d_i >_c d_j \mid \mathbf{m})(\phi(d_i, d_j, \mathbf{r}_n, \mathbf{m}) - \phi(d_i, d_j, \mathbf{r}_m, \mathbf{m})) \\ + P(d_j >_c d_i \mid \mathbf{m})(\phi(d_j, d_i, \mathbf{r}_n, \mathbf{m}) - \phi(d_j, d_i, \mathbf{r}_m, \mathbf{m}))] \tag{2.25}$$

will be zero if $|A| = |B|$ and positive if $|A| < |B|$ under correlated clicks. Moreover, since $\mathbf{r}_n$ Pareto dominates $\mathbf{r}_m$, there will be at least one document $d_j$ where:

$$\exists d_i, \exists d_j, rel(d_i) \wedge \neg rel(d_j) \wedge r(d_i, \mathbf{l}_n) = r(d_j, \mathbf{l}_m). \tag{2.26}$$

This means that the expected outcome (Equation 2.15) will always be positive under correlated clicks, i.e., $E[P_{nm}] > 0$, for a Pareto dominating ranker $\mathbf{r}_n$ and any other ranker $\mathbf{r}_m$.

In summary, we have introduced a new multileaved comparison method, PPM. Furthermore, we answered **RQ2.1** in the affirmative since we have shown it to be *considerate* and to have *fidelity*. We further note that PPM has polynomial complexity: to calculate $P(\underline{r}(i, j, \mathbf{m}) \geq \bar{r}(i, j))$ only the size of the choice sets $\Omega$ and the first positions at which $d_i$ and $d_j$ occur in $\Omega$ have to be known.

## 2.6 Experiments

In order to answer Research Question **RQ2.2** posed in Section 2.1 several experiments were performed to evaluate the *sensitivity* of PPM. The methodology of evaluation follows previous work on interleaved and multileaved comparison methods [12, 41, 44, 108, 109] and is completely reproducible.

### 2.6.1 Ranker selection and comparisons

In order to make fair comparisons between rankers, we will use the LTR datasets described in Section 2.6.2 below. From the feature representations in these datasets a handpicked set of features was taken and used as ranking models. To match the real-world scenario as best as possible this selection consists of features that are known to perform well as relevance signals independently. This selection includes but is not limited to: BM25, LMIR.JM, Sitemap, PageRank, HITS and TF.IDF [108].

Then the ground-truth comparisons between the rankers are based on their NDCG scores computed on a held-out test set, resulting in a binary preference matrix $P_{nm}$ for all ranker pairs $(\mathbf{r}_n, \mathbf{r}_m)$:

$$P_{nm} = NDCG(\mathbf{r}_n) - NDCG(\mathbf{r}_m). \tag{2.27}$$

The metric by which multileaved comparison methods are compared is the *binary error*, $E_{bin}$ [12, 108, 109]. Let $\hat{P}_{nm}$ be the preference inferred by a multileaved comparison method; then the error is:

$$E_{bin} = \frac{\sum_{n,m \in \mathcal{R} \wedge n \neq m} \mathrm{sgn}(\hat{P}_{nm}) \neq sgn(P_{nm})}{|\mathcal{R}| \times (|\mathcal{R}| - 1)}. \tag{2.28}$$

### 2.6.2 Datasets

Our experiments are performed over ten publicly available LTR datasets with varying sizes and representing different search tasks. Each dataset consists of a set of queries

Table 2.2: Instantiations of Cascading Click Models [36] as used for simulating user behaviour in experiments.

| | $P(click = 1 \mid R)$ | | | | | $P(stop = 1 \mid R)$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $R$ | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | 4 |
| *perfect* | 0.0 | 0.2 | 0.4 | 0.8 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| *navigational* | 0.05 | 0.3 | 0.5 | 0.7 | 0.95 | 0.2 | 0.3 | 0.5 | 0.7 | 0.9 |
| *informational* | 0.4 | 0.6 | 0.7 | 0.8 | 0.9 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |

and a set of corresponding documents for every query. While queries are represented only by their identifiers, feature representations and relevance labels are available for every document-query pair. Relevance labels are graded differently by the datasets depending on the task they model, for instance, navigational datasets have binary labels for not relevant (0), and relevant (1), whereas most informational tasks have labels ranging from not relevant (0), to perfect relevancy (4). Every dataset consists of five folds, each dividing the dataset in different training, validation and test partitions.

The first publicly available LTR datasets are distributed as LETOR 3.0 and 4.0 [76]; they use representations of 45, 46, or 64 features encoding ranking models such as TF.IDF, BM25, Language Modelling, PageRank, and HITS on different parts of the documents. The datasets in LETOR are divided by their tasks, most of which come from the TREC Web Tracks between 2003 and 2008 [23, 24]. *HP2003, HP2004, NP2003, NP2004, TD2003* and *TD2004* each contain between 50 and 150 queries and 1,000 judged documents per query and use binary relevance labels. Due to their similarity we report average results over these six datasets noted as *LETOR 3.0*. The *OHSUMED* dataset is based on the query log of the search engine on the MedLine abstract database, and contains 106 queries. The last two datasets, *MQ2007* and *MQ2008*, were based on the Million Query Track [8] and consist of 1,700 and 800 queries, respectively, but have far fewer assessed documents per query.

The *MLSR-WEB10K* dataset [95] consists of 10,000 queries obtained from a retired labelling set of a commercial web search engine. The datasets uses 136 features to represent its documents, each query has around 125 assessed documents.

Finally, we note that there are more LTR datasets that are publicly available [17, 27], but there is no public information about their feature representations. Therefore, they are unfit for our evaluation as no selection of well performing ranking features can be made.

## 2.6.3   Simulating user behavior

While experiments using real users are preferred [18, 21, 63, 133], most researchers do not have access to search engines. As a result the most common way of comparing online evaluation methods is by using simulated user behaviour [12, 41, 44, 108, 109]. Such simulated experiments show the performance of multileaved comparison methods when user behaviour adheres to a few simple assumptions.

Our experiments follow the precedent set by previous work on online evaluation:

First, a user issues a query simulated by uniformly sampling a query from the static dataset. Subsequently, the multileaved comparison method constructs the multileaved result list of documents to display. The behavior of the user after receiving this list is simulated using a *cascade click model* [20, 36]. This model assumes a user to examine documents in their displayed order. For each document that is considered the user decides whether it warrants a click, which is modeled as the conditional probability $P(click = 1 \mid R)$ where $R$ is the relevance label provided by the dataset. Accordingly, *cascade click model* instantiations increase the probability of a click with the degree of the relevance label. After the user has clicked on a document their information need may be satisfied; otherwise they continue considering the remaining documents. The probability of the user not examining more documents after clicking is modeled as $P(stop = 1 \mid R)$, where it is more likely that the user is satisfied from a very relevant document. At each impression we display $k = 10$ documents to the user.

Table 2.2 lists the three instantiations of cascade click models that we use for this chapter. The first models a *perfect* user who considers every document and clicks on all relevant documents and nothing else. Secondly, the *navigational* instantiation models a user performing a navigational task who is mostly looking for a single highly relevant document. Finally, the *informational* instantiation models a user without a very specific information need who typically clicks on multiple documents. These three models have increasing levels of noise, as the behavior of each depends less on the relevance labels of the displayed documents.

### 2.6.4 Experimental runs

Each experimental run consists of applying a multileaved comparison method to a sequence of $T = 10,000$ simulated user impressions. To see the effect of the number of rankers in a comparison, our runs consider $|\mathcal{R}| = 5$, $|\mathcal{R}| = 15$, and $|\mathcal{R}| = 40$. However only the *MSLR* dataset contains $|\mathcal{R}| = 40$ rankers. Every run is repeated for every click model to see how different behaviours affect performance. For statistical significance every run is repeated 25 times per fold, which means that 125 runs are conducted for every dataset and click model pair. Since our evaluation covers five multileaved comparison methods, we generate over 393 million impressions in total. We test for statistical significant differences using a two tailed t-test. Note that the results reported on the LETOR 3.0 data are averaged over six datasets and thus span 750 runs per datapoint.

The parameters of the baselines are selected based on previous work on the same datasets; for OM the sample size $\eta = 10$ was chosen as reported by Schuth et al. [108]; for PM the degree $\tau = 3.0$ was chosen according to Hofmann et al. [41] and the sample size $\eta = 10,000$ in accordance with Schuth et al. [109].

## 2.7 Results and Analysis

We answer Research Question **RQ2.2** by evaluating the *sensitivity* of PPM based on the results of the experiments detailed in Section 2.6.

The results of the experiments with a smaller number of rankers: $|\mathcal{R}| = 5$ are

Table 2.3: The binary error $E_{bin}$ of all multileaved comparison methods after 10,000 impressions on comparisons of $|\mathcal{R}| = 5$ rankers. Average per dataset and click model; standard deviation in brackets. The best performance per click model and dataset is noted in bold, statistically significant improvements of PPM are noted by $^{\blacktriangledown}(p < 0.01)$ and $^{\triangledown}(p < 0.05)$ and losses by $^{\blacktriangle}$ and $^{\triangle}$ respectively or $\circ$ for no difference, per baseline.

| | TDM | OM | PM | SOSM | PPM |
|---|---|---|---|---|---|
| | | | *perfect* | | |
| LETOR 3.0 | 0.16 ( 0.13) | **0.14** ( 0.15) | 0.15 ( 0.15) | 0.16 ( 0.15) | **0.14** ( 0.13) ∘∘∘∘ |
| MQ2007 | 0.19 ( 0.16) | 0.22 ( 0.18) | **0.16** ( 0.14) | 0.18 ( 0.16) | **0.16** ( 0.14) ∘▼∘∘ |
| MQ2008 | 0.15 ( 0.12) | 0.19 ( 0.14) | 0.16 ( 0.12) | 0.18 ( 0.15) | **0.14** ( 0.12) ∘▼∘▽ |
| MSLR-WEB10k | 0.23 ( 0.13) | 0.27 ( 0.17) | 0.20 ( 0.14) | 0.25 ( 0.18) | **0.14** ( 0.13) ▼▼▼▼ |
| OHSUMED | 0.14 ( 0.12) | 0.19 ( 0.15) | **0.11** ( 0.09) | **0.11** ( 0.10) | **0.11** ( 0.10) ▼▼∘∘ |
| | | | *navigational* | | |
| LETOR 3.0 | 0.16 ( 0.13) | **0.15** ( 0.15) | **0.15** ( 0.14) | 0.17 ( 0.15) | 0.16 ( 0.14) ∘∘∘∘ |
| MQ2007 | 0.21 ( 0.17) | 0.33 ( 0.21) | 0.18 ( 0.12) | 0.29 ( 0.23) | **0.17** ( 0.14) ∘▼∘▼ |
| MQ2008 | 0.17 ( 0.14) | 0.21 ( 0.20) | 0.17 ( 0.15) | 0.23 ( 0.18) | **0.15** ( 0.13) ∘▼∘▼ |
| MSLR-WEB10k | 0.24 ( 0.14) | 0.32 ( 0.20) | 0.24 ( 0.17) | 0.31 ( 0.19) | **0.20** ( 0.15) ▽▼▽▼ |
| OHSUMED | **0.12** ( 0.11) | 0.27 ( 0.19) | 0.14 ( 0.12) | 0.23 ( 0.17) | 0.13 ( 0.12) ∘▼∘▼ |
| | | | *informational* | | |
| LETOR 3.0 | 0.16 ( 0.14) | 0.22 ( 0.19) | **0.14** ( 0.11) | 0.17 ( 0.15) | 0.15 ( 0.13) ∘▼∘∘ |
| MQ2007 | 0.23 ( 0.15) | 0.41 ( 0.26) | 0.23 ( 0.15) | 0.37 ( 0.23) | **0.17** ( 0.16) ▼▼▼▼ |
| MQ2008 | 0.18 ( 0.13) | 0.28 ( 0.19) | 0.18 ( 0.16) | 0.23 ( 0.18) | **0.17** ( 0.14) ∘▼∘▼ |
| MSLR-WEB10k | 0.27 ( 0.18) | 0.42 ( 0.23) | 0.24 ( 0.17) | 0.36 ( 0.20) | **0.19** ( 0.17) ▼▼▽▼ |
| OHSUMED | 0.13 ( 0.10) | 0.37 ( 0.24) | **0.12** ( 0.11) | 0.27 ( 0.21) | **0.12** ( 0.10) ∘▼∘▼ |

displayed in Table 2.3. Here we see that after 10,000 impressions PPM has a significantly lower error on many datasets and at all levels of interaction noise. Furthermore, for $|\mathcal{R}| = 5$ there are no significant losses in performance under any circumstances.

When $|\mathcal{R}| = 15$ as displayed in Table 2.4, we see a single case where PPM performs worse than a previous method: on *MQ2007* under the *perfect* click model SOSM performs significantly better than PPM. However, on the same dataset PPM performs significantly better under the *informational* click model. Furthermore, there are more significant improvements for $|\mathcal{R}| = 15$ than when the number of rankers is the smaller $|\mathcal{R}| = 5$.

Finally, when the number of rankers in the comparison is increased to $|\mathcal{R}| = 40$ as displayed in Table 2.5, PPM still provides significant improvements.

We conclude that PPM, in the experimental conditions that we considered, provides a performance that is at least as good as any existing method. Moreover, PPM is robust to noise as we see more significant improvements under click-models with increased noise. Furthermore, since improvements are found with the number of rankers $|\mathcal{R}|$ varying from 5 to 40, we conclude that PPM is scalable in the comparison size. Additionally, the

Table 2.4: The binary error $E_{bin}$ after 10,000 impressions on comparisons of $|\mathcal{R}| = 15$ rankers. Notation is identical to Table 2.3.

|  | TDM | OM | PM | SOSM | PPM |
|---|---|---|---|---|---|
| | | | *perfect* | | |
| LETOR 3.0 | 0.16 ( 0.07) | **0.14** ( 0.08) | 0.15 ( 0.07) | 0.17 ( 0.08) | 0.16 ( 0.08) ○○○○ |
| MQ2007 | 0.20 ( 0.07) | 0.25 ( 0.09) | 0.18 ( 0.06) | **0.15** ( 0.07) | 0.19 ( 0.07) ○▼○▲ |
| MQ2008 | 0.16 ( 0.05) | 0.17 ( 0.05) | 0.16 ( 0.05) | **0.15** ( 0.07) | **0.15** ( 0.06) ○▽○○ |
| MSLR-WEB10k | 0.24 ( 0.07) | 0.38 ( 0.11) | 0.21 ( 0.06) | 0.30 ( 0.08) | **0.14** ( 0.05) ▼▼▼▼ |
| OHSUMED | 0.14 ( 0.03) | 0.18 ( 0.05) | 0.13 ( 0.03) | 0.13 ( 0.03) | **0.11** ( 0.03) ▼▼▼▼ |
| | | | *navigational* | | |
| LETOR 3.0 | 0.16 ( 0.08) | 0.16 ( 0.09) | **0.15** ( 0.08) | 0.17 ( 0.08) | 0.17 ( 0.08) ○○○○ |
| MQ2007 | 0.24 ( 0.07) | 0.33 ( 0.11) | **0.20** ( 0.07) | 0.22 ( 0.08) | 0.21 ( 0.08) ▼▼○○ |
| MQ2008 | 0.19 ( 0.05) | 0.21 ( 0.07) | **0.16** ( 0.05) | 0.18 ( 0.06) | **0.16** ( 0.06) ▼▼○▼ |
| MSLR-WEB10k | 0.27 ( 0.07) | 0.42 ( 0.12) | 0.24 ( 0.06) | 0.28 ( 0.09) | **0.22** ( 0.08) ▼▼○▼ |
| OHSUMED | 0.14 ( 0.04) | 0.25 ( 0.07) | **0.13** ( 0.03) | 0.18 ( 0.06) | 0.13 ( 0.04) ○▼○▼ |
| | | | *informational* | | |
| LETOR 3.0 | 0.18 ( 0.07) | 0.20 ( 0.11) | 0.17 ( 0.08) | **0.16** ( 0.08) | 0.18 ( 0.08) ○▽○○ |
| MQ2007 | 0.28 ( 0.07) | 0.42 ( 0.14) | 0.26 ( 0.08) | 0.28 ( 0.11) | **0.21** ( 0.08) ▼▼▼▼ |
| MQ2008 | 0.23 ( 0.06) | 0.26 ( 0.11) | 0.18 ( 0.06) | 0.20 ( 0.06) | **0.15** ( 0.06) ▼▼▼▼ |
| MSLR-WEB10k | 0.30 ( 0.09) | 0.45 ( 0.12) | 0.28 ( 0.08) | 0.35 ( 0.11) | **0.24** ( 0.08) ▼▼▼▼ |
| OHSUMED | 0.15 ( 0.03) | 0.42 ( 0.09) | **0.13** ( 0.03) | 0.25 ( 0.06) | 0.13 ( 0.04) ▼▼○▼ |

dataset type seems to affect the relative performance of the methods. For instance, on *LETOR 3.0* little significant differences are found, whereas the *MSLR* dataset displays the most significant improvements. This suggests that on more artificial data, i.e., the smaller datasets simulating navigational tasks, the differences are fewer, while on the other hand on large commercial data the preference for PPM increases further. Lastly, Figure 2.1 displays the binary error of all multileaved comparison methods on the *MSLR* dataset over 10,000 impressions. Under the *perfect* click model we see that all of the previous methods display converging behavior around 3,000 impressions. In contrast, the error of PPM continues to drop throughout the experiment. The fact that the existing methods converge at a certain level of error in the absence of click-noise is indicative that they are lacking in *sensitivity*.

Overall, our results show that PPM reaches a lower level of error than previous methods seem to be capable of. This feat can be observed on a diverse set of datasets, various levels of interaction noise and for different comparison sizes. To answer Research Question **RQ2.2**: from our results we conclude that PPM is more sensitive than any existing multileaved comparison method.

Table 2.5: The binary error $E_{bin}$ of all multileaved comparison methods after 10,000 impressions on comparisons of $|\mathcal{R}| = 40$ rankers. Averaged over the *MSLR-WEB10k*, notation is identical to Table 2.3.

|  | TDM | OM | PM | SOSM | PPM |
|---|---|---|---|---|---|
| *perfect* | 0.26 $_{(0.03)}$ | 0.43 $_{(0.02)}$ | 0.23 $_{(0.02)}$ | 0.31 $_{(0.02)}$ | **0.18** $_{(0.04)}$ ▼▼▼▼ |
| *navigational* | 0.31 $_{(0.03)}$ | 0.44 $_{(0.01)}$ | 0.25 $_{(0.03)}$ | **0.23** $_{(0.03)}$ | 0.24 $_{(0.05)}$ ▼▼▼○ |
| *informational* | 0.37 $_{(0.04)}$ | 0.47 $_{(0.01)}$ | 0.30 $_{(0.05)}$ | 0.34 $_{(0.05)}$ | **0.27** $_{(0.06)}$ ▼▼▼▼ |

## 2.8 Conclusion

In this chapter we have examined multileaved comparison methods for evaluating ranking models online.

We have presented a new multileaved comparison method, Pairwise Preference Multileaving (PPM), that is more sensitive to user preferences than existing methods. Additionally, we have proposed a theoretical framework for assessing multileaved comparison methods, with *considerateness* and *fidelity* as the two key requirements. We have shown that no method published prior to PPM has *fidelity* without lacking *considerateness*. In other words, prior to PPM no multileaved comparison method has been able to infer correct preferences without degrading the search experience of the user. In contrast, we prove that PPM has both *considerateness* and *fidelity*, thus it is guaranteed to correctly identify a Pareto dominating ranker without altering the search experience considerably. Furthermore, our experimental results spanning ten datasets show that PPM is more sensitive than existing methods, meaning that it can reach a lower level of error than any previous method. Moreover, our experiments show that the most significant improvements are obtained on the more complex datasets, i.e., larger datasets with more grades of relevance. Additionally, similar improvements are observed under different levels of noise and numbers of rankers in the comparison, indicating that PPM is robust to interaction noise and scalable to large comparisons. As an extra benefit, the computational complexity of PPM is polynomial and, unlike previous methods, does not depend on sampling or approximations.

With these findings we can answer the thesis research question **RQ1** positively: with the introduction of our novel Pairwise Preference Multileaving (PPM) method the effectiveness of online evaluation scales to large comparisons.

The theoretical framework that we have introduced allows future research into multileaved comparison methods to guarantee improvements that generalize better than empirical results alone. In turn, properties like *considerateness* can further stimulate the adoption of multileaved comparison methods in production environments; future work with real-world users may yield further insights into the effectiveness of the multileaving paradigm. Rich interaction data enables the introduction of multileaved comparison methods that consider more than just clicks, as has been done for interleaving methods [63]. These methods could be extended to consider other signals such as *dwell-time* or *the order of clicks in an impression*, etc.

Furthermore, the field of Online Learning to Rank (OLTR) has depended on online

evaluation from its inception [132]. The introduction of multileaving and subsequent novel multileaved comparison methods brought substantial improvements to both fields [90, 111]. Similarly, PPM and any future extensions are likely to benefit the OLTR field too.

Finally, while the theoretical and empirical improvements of PPM are convincing, future work should investigate whether the sensitivity can be made even stronger. For instance, it is possible to have clicks from which no preferences between rankers can be inferred. Can we devise a method that avoids such situations as much as possible without introducing any form of bias, thus increasing the sensitivity even further while maintaining theoretical guarantees?

In Chapter 7 we will take another look at online ranker evaluation and contrast it with counterfactual evaluation. We will see that existing interleaving methods (and by extension some multileaving methods) are biased w.r.t. the definition of position bias common in counterfactual evaluation. The novel method introduced in Chapter 7 combines aspects of counterfactual and online ranker evaluation, creating a method with strong theoretical guarantees while also being very effective.

Furthermore, similar to this chapter, Chapter 3 will look at whether a pairwise LTR method is suitable for online LTR. While different from PPM, the method introduced in Chapter 3 also infers pairwise preferences between documents, and weights inferred preferences to account for position bias.
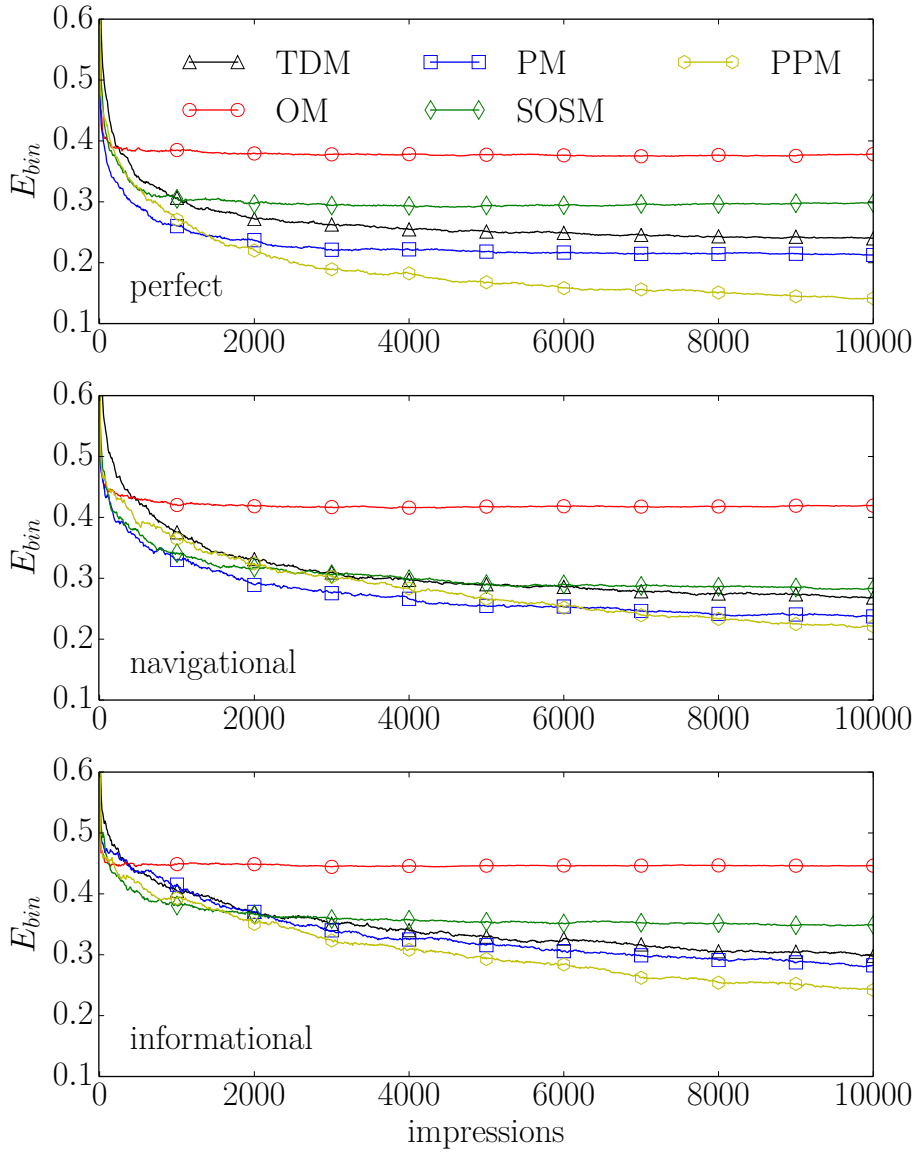
Figure 2.1: The binary error of different multileaved comparison methods on comparisons of $|\mathcal{R}| = 15$ rankers on the *MSLR-WEB10k* dataset.

## 2.A   Notation Reference for Chapter 2

| Notation | Description |
|---|---|
| $q$ | a user-issued query |
| $T$ | the total number of interactions |
| $\mathbf{r}_i$ | an individual ranker a.k.a. a single ranking system or ranking model |
| $\mathcal{R}$ | a set of rankers to compare |
| $\mathbf{l}_i$ | a ranking generated by ranker $\mathbf{r}_i$ |
| $\mathbf{m}$ | a multileaved result list |
| $k$ | the length of the multileaved result lists |
| $\mathbf{c}$ | a vector indicating clicks on a displayed multileaved result list |
| $P$ | a preference matrix to store inferred preferences between rankers |
| $r(d, \mathbf{l}_i)$ | the rank at which ranker $\mathbf{r}_i$ places document $d$ |

# 3

# Differentiable Online Learning to Rank

Online Learning to Rank (OLTR) methods optimize rankers based on direct interaction with users. State-of-the-art OLTR methods rely on online evaluation and sampling model variants, they were designed specifically for linear models. Their approaches do not extend well to non-linear models such as neural networks.

To address this limitation, this chapter will consider the thesis research question:

**RQ2** *Is online Learning to Rank (LTR) possible without relying on model-sampling and online evaluation?*

We introduce an entirely novel approach to OLTR that constructs a weighted differentiable pairwise loss after each interaction: Pairwise Differentiable Gradient Descent (PDGD). PDGD breaks away from the traditional approach that relies on interleaving or multileaving and extensive sampling of models to estimate gradients. Instead, its gradient is based on inferring preferences between document pairs from user clicks and can optimize any differentiable model. We prove that the gradient of PDGD is unbiased w.r.t. user document pair preferences. Our experiments on the largest publicly available LTR datasets show considerable and significant improvements under all levels of interaction noise. PDGD outperforms existing OLTR methods both in terms of learning speed as well as final convergence. Furthermore, unlike previous OLTR methods, PDGD also allows for non-linear models to be optimized effectively. Our results show that using a neural network leads to even better performance at convergence than a linear model. In summary, PDGD is an efficient and unbiased OLTR approach that provides a better user experience than previously possible.

## 3.1 Introduction

In order to benefit from unprecedented volumes of content, users rely on ranking systems to provide them with the content of their liking. LTR in Information Retrieval (IR) concerns methods that optimize ranking models so that they order documents according to user preferences. In web search engines such models combine hundreds of signals to rank web-pages according to their relevance to user queries [75]. Similarly, ranking models are a vital part of recommender systems where there is no explicit search intent

---

This chapter was published as [82]. Appendix 3.A gives a reference for the notation used in this chapter.

[59]. LTR is also prevalent in settings where other content is ranked, e.g., videos [19], products [60], conversations [97] or personal documents [127].

Traditionally, LTR has been applied in the *offline* setting where a dataset with annotated query-document pairs is available. Here, the model is optimized to rank documents according to the relevance annotations, which are based on the judgements of human annotators. Over time the limitations of this supervised approach have become apparent: annotated sets are expensive and time-consuming to create [17, 76]; when personal documents are involved such a dataset would breach privacy [127]; the relevance of documents to queries can change over time, like in a news search engine [1, 71]; and judgements of raters are not necessarily aligned with the actual users [104].

In order to overcome the issues with annotated datasets, previous work in LTR has looked into learning from user interactions. Work along these lines can be divided into *approaches that learn from historical interactions*, i.e., in the form of interaction logs [54], and *approaches that learn in an online setting* [132]. The latter regard methods that determine what to display to the user at each impression, and then immediately learn from observed user interactions and update their behavior accordingly. This online approach has the advantage that it does not require an existing ranker of decent quality, and thus can handle cold-start situations. Additionally, it is more responsive to the user by updating continuously and instantly, therefore allowing for a better experience. However, it is important that an online method can handle biases that come with user behavior: for instance, the observed interactions only take place with the displayed results, i.e., there is item-selection bias, and are more likely to occur with higher ranked items, i.e., there is position bias. Accordingly, a method should learn user preferences w.r.t. document relevance, and be robust to the forms of noise and bias present in the online setting. Overall, the online LTR approach promises to learn ranking models that are in line with user preferences, in a responsive matter, reaching good performance from few interactions, even in cold-start situations.

Despite these highly beneficial properties, previous work in OLTR has only considered linear models [42, 111, 132] or trivial variants thereof [80]. The reason for this is that existing work in OLTR has worked with the Dueling Bandit Gradient Descent (DBGD) algorithm [132] as a basis. While very influential and effective, we identify two main problems with the gradient estimation of the DBGD algorithm:

1. Gradient estimation is based on sampling model variants from a unit circle around the current model. This concept does not extend well to non-linear models. Computing rankings for variants is also computationally costly for larger complex models.

2. It uses online evaluation methods, i.e., interleaving or multileaving, to determine the gradient direction from the resulting set of models. However, these evaluation methods are designed for finding preferences between ranking systems, not (primarily) for determining how a model should be updated.

As an alternative we introduce *Pairwise Differentiable Gradient Descent* (PDGD), the first unbiased OLTR method that is applicable to any differentiable ranking model. PDGD infers pairwise document preferences from user interactions and constructs an unbiased gradient after each user impression. In addition, PDGD does not rely on sam-

pling models for exploration, but instead models rankings as probability distributions over documents. Therefore, it allows the OLTR model to be very certain for specific queries and perform less exploration in those cases, while being much more explorative in other, uncertain cases. Our results show that, consequently, PDGD provides significant and considerable improvements over previous OLTR methods. This indicates that its gradient estimation is more in line with the preferences to be learned.

In this chapter, we address the thesis research question **RQ2** by answering the following three specific research questions:

**RQ3.1** Does using PDGD result in significantly better performance than the current state-of-the-art Multileave Gradient Descent?

**RQ3.2** Is the gradient estimation of PDGD unbiased?

**RQ3.3** Is PDGD capable of effectively optimizing different types of ranking models?

To facilitate replicability and repeatability of our findings, we provide open source implementations of PDGD and our experiments under the permissive MIT open-source license.[1]

## 3.2   Related Work

### 3.2.1   Learning to rank

LTR can be applied to the offline and online setting. In the offline setting LTR is approached as a supervised problem where the relevance of each query-document pair is known. Most of the challenges with offline LTR come from obtaining annotations. For instance, gathering annotations is time-consuming and expensive [17, 76, 95]. Furthermore, in privacy sensitive-contexts it would be unethical to annotate items, e.g., for personal emails or documents [127]. Moreover, for personalization problems annotators are unable to judge what specific users would prefer. Also, (perceived) relevance chances over time, due to cognitive changes on the user's end [120] or due to changes in document collections [1] or the real world [71]. Finally, annotations are not necessarily aligned with user satisfaction, as judges may interpret queries differently from actual users [104]. Consequently, the limitations of offline LTR have led to an increased interest in alternative approaches to LTR.

### 3.2.2   Online learning to rank

OLTR is an attractive alternative to offline LTR as it learns directly from interacting with users [132]. By doing so it attempts to solve the issues with offline annotations that occur in LTR, as user preferences are expected to be better represented by interactions than by offline annotations [99]. Unlike methods in the offline setting, OLTR algorithms have to simultaneously perform ranking while also optimizing their ranking model. In other words, an OLTR algorithm decides what rankings to display to users, while at

---

[1] https://github.com/HarrieO/OnlineLearningToRank

the same time learning from the interactions with the presented rankings. While the potential of learning in the online setting is great, it has its own challenges. In particular, the main difficulties of the OLTR task are *bias* and *noise*. Any user interaction that does not reflect their true preference is considered noise, this happens frequently e.g., clicks often occur for unexpected reasons [104]. Bias comes in many forms, for instance, item-selection bias occurs because interactions only involve displayed documents [127]. Another common bias is position bias, a consequence from the fact documents at the top of a ranking are more likely to be considered [134]. An OLTR method should thus take into account the biases that affect user behavior while also being robust to noise, in order to learn the *true* user preferences.

OLTR methods can be divided into two groups [139]: *tabular* methods that learn the best ranked list under some model of user interaction with the list [98, 114], such as a click model [20], and *feature-based* algorithms that learn the best ranker in a family of rankers [43, 132]. Model-based methods may have greater statistical efficiency but they give up generality, essentially requiring us to learn a separate model for every query. For the remainder of this chapter, we focus on model-free OLTR methods.

### 3.2.3 DBGD and beyond

State-of-the-art (model-free) OLTR approaches learn user preferences by approaching optimization as a dueling bandit problem [132]. They estimate the gradient of the model w.r.t. user satisfaction by comparing the current model to sampled variations of the model. The original DBGD algorithm [132] uses interleaving methods to make these comparisons: at each interaction the rankings of two rankers are combined to create a single result list. From a large number of clicks on such a combined result list a user preference between the two rankers can reliably be inferred [41]. Conversely, DBGD compares its current ranking model to a different slight variation at each impression. Then, if a click is indicative of a preference for the variation, the current model is slightly updated towards it. Accordingly, the model of DBGD will continuously update itself and oscillate towards an inferred optimum.

Other work in OLTR has used DBGD as a basis and extended upon it. Notably, Hofmann et al. [43] have proposed a method that guides exploration by only sampling variations that seem promising from historical interaction data. Unfortunately, while this approach provides faster initial learning, the historical data introduces bias which leads to the quality of the ranking model to steadily decrease over time [90]. Alternatively, Schuth et al. [111] introduced Multileave Gradient Descent (MGD), this extension replaced the interleaving of DBGD with multileaving methods. In turn the multileaving paradigm is an extension of interleaving where a set of rankers are compared efficiently [81, 108, 109]. Conversely, multileaving methods can combine the rankings of more than two rankers and thus infer preferences over a set of rankers from a single click. MGD uses this property to estimate the gradient more effectively by comparing a large number of model variations per user impression [90, 111]. As a result, MGD requires fewer user interactions to converge on the same level of performance as DBGD. Another alternative approach was considered by Hofmann et al. [40], who inject the ranking from the current model with randomly sampled documents. Then, after each user impression, a pairwise loss is constructed from inferred preferences

between documents. This pairwise approach was not found to be more effective than DBGD.

Quite remarkably, all existing work in OLTR has only considered linear models. Recently, Oosterhuis and de Rijke [80] recognized that a tradeoff unique to OLTR arises when choosing models. High capacity models such as neural networks [13] require more data than simpler models. On the one hand, this means that high capacity models need more user interactions to reach the same level of performance, thus giving a worse initial user experience. On the other hand, high capacity models are capable of finding better optima, thus lead to better final convergence and a better long-term user experience. This dilemma is named the *speed-quality* tradeoff, and as a solution a cascade of models can be optimized: combining the initial learning speed of a simple model with the convergence of a complex one. But there are more reasons why non-linear models have so far been absent from OLTR. Importantly, the DBGD algorithm was designed for linear models from the ground up; relying on a unit circle to sample model variants and averaging models to estimate the gradient. Furthermore, the computational cost of maintaining an extensive set of model variants for large and complex models makes this approach very impractical.

Our contribution over the work listed above is an OLTR method that is not an extension of DBGD, instead it computes a differentiable pairwise loss to update its model. Unlike the existing pairwise approach, our loss function is unbiased and our exploration is performed using the model's confidence over documents. Finally, we also show that this is the first OLTR method to effectively optimize neural networks in the online setting.

## 3.3  Method

In this section we introduce a novel OLTR algorithm: Pairwise Differentiable Gradient Descent (PDGD). First, Section 3.3.1 describes PDGD in detail, before Section 3.3.2 formalizes and proves the unbiasedness of the method. Appendix 3.A lists the notation we use.

### 3.3.1  Pairwise Differentiable Gradient Descent

PDGD revolves around optimizing a ranking model $f_\theta(\mathbf{d})$ that takes a feature representation of a query-document pair $\mathbf{d}$ as input and outputs a score. The aim of the algorithm is to find the parameters $\theta$ so that sorting the documents by their scores in descending order provides the most optimal rankings. Because this is an online algorithm, the method must first decide what ranking to display to the user, then after the user has interacted with the displayed ranking, it may update $\theta$ accordingly.

Unlike previous OLTR approaches, PDGD does not rely on any online evaluation methods. Instead, a Plackett-Luce (PL) model is applied to the ranking function $f_\theta(\cdot)$ resulting in a distribution over the document set $D$:

$$P(d|D) = \frac{e^{f_\theta(\mathbf{d})}}{\sum_{d' \in D} e^{f_\theta(\mathbf{d'})}}. \tag{3.1}$$
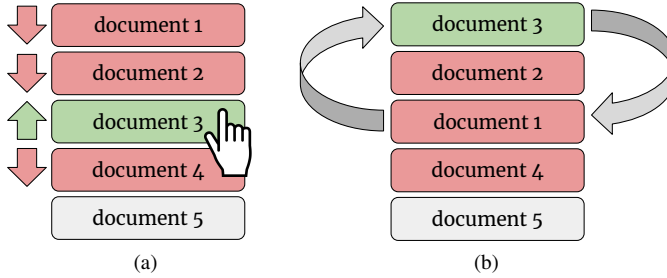
Figure 3.1: Left: a click on a document ranking $R$ and the inferred preferences of $d_3$ over $\{d_1, d_2, d_4\}$. Right: the reversed pair ranking $R^*(d_1, d_3, R)$ for the document pair $d_1$ and $d_3$.

A ranking $R$ to display to the user is then created by sampling from the distribution $k$ times, where after each placement the distribution is renormalized to prevent duplicate placements. PL models have been used before in LTR. For instance, the ListNet method [15] optimizes such a model in the offline setting. With $R_i$ denoting the document at position $i$, the probability of the ranking $R$ then becomes:

$$P(R|D) = \prod_{i=1}^{k} P(R_i | D \setminus \{R_1, \ldots, R_{i-1}\}). \tag{3.2}$$

After the ranking $R$ has been displayed to the user, they have the option to interact with it. The user may choose to click on some (or none) of the documents. Based on these clicks, PDGD will infer preferences between the displayed documents. We assume that clicked documents are preferred over observed unclicked documents. However, to the algorithm it is unknown which unclicked documents the user has considered. As a solution, PDGD relies on the assumption that every document preceding a clicked document and the first subsequent unclicked document was observed, as illustrated in Figure 3.1a. This preference assumption has been proven useful in IR before, for instance in pairwise LTR on click logs [54] and recently in online evaluation [81]. We will denote preferences between documents inferred from clicks as: $d_k >_{\mathbf{c}} d_l$ where $d_k$ is preferred over $d_l$.

Then $\theta$ is updated by optimizing pairwise probabilities over the preference pairs; for each inferred document preference $d_k >_{\mathbf{c}} d_l$, the probability that the preferred document $d_k$ is sampled before $d_l$ is sampled is increased [118]:

$$P(d_k \succ d_l) = \frac{P(d_k|D)}{P(d_k|D) + P(d_l|D)} = \frac{e^{f(\mathbf{d}_k)}}{e^{f(\mathbf{d}_k)} + e^{f(\mathbf{d}_l)}}. \tag{3.3}$$

We have chosen for pairwise optimization over listwise optimization because a pairwise method can be made unbiased by reweighing preference pairs. To do this we introduce the weighting function $\rho(d_k, d_l, R, D)$ and estimate the gradient of the user preferences

**Algorithm 3.1** Pairwise Differentiable Gradient Descent (PDGD).

---

1: **Input**: initial weights: $\theta_1$; scoring function: $f$; learning rate $\eta$.
2: **for** $t \leftarrow 1 \ldots \infty$ **do**
3:      $q_t \leftarrow \text{receive\_query}(t)$      *// obtain a query from a user*
4:      $D_t \leftarrow \text{preselect\_documents}(q_t)$      *// preselect documents for query*
5:      $\mathbf{R}_t \leftarrow \text{sample\_list}(f_{\theta_t}, D_t)$      *// sample list according to Eq. 3.1*
6:      $\mathbf{c}_t \leftarrow \text{receive\_clicks}(\mathbf{R}_t)$      *// show result list to the user*
7:      $\nabla f_{\theta_t} \leftarrow \mathbf{0}$      *// initialize gradient*
8:      **for** $d_k >_{\mathbf{c}} d_l \in \mathbf{c}_t$ **do**
9:          $w \leftarrow \rho(d_k, d_l, R, D)$      *// initialize pair weight (Eq. 3.5)*
10:         $w \leftarrow w \frac{e^{f_{\theta_t}(\mathbf{d}_k)} e^{f_{\theta_t}(\mathbf{d}_l)}}{\left(e^{f_{\theta_t}(\mathbf{d}_k)} + e^{f_{\theta_t}(\mathbf{d}_l)}\right)^2}$      *// pair gradient (Eq. 3.4)*
11:         $\nabla f_{\theta_t} \leftarrow \nabla \theta_t + w(f'_{\theta_t}(\mathbf{d}_k) - f'_{\theta_t}(\mathbf{d}_l))$      *// model gradient (Eq. 3.4)*
12:      $\theta_{t+1} \leftarrow \theta_t + \eta \nabla f_{\theta_t}$      *// update the ranking model*

---

by the weighted sum:

$$
\begin{aligned}
\nabla f_\theta(\cdot) &\approx \sum_{d_k >_{\mathbf{c}} d_l} \rho(d_k, d_l, R, D) \left[\nabla P(d_k \succ d_l)\right] \\
&= \sum_{d_k >_{\mathbf{c}} d_l} \rho(d_k, d_l, R, D) \frac{e^{f_\theta(\mathbf{d}_k)} e^{f_\theta(\mathbf{d}_l)}}{(e^{f_\theta(\mathbf{d}_k)} + e^{f_\theta(\mathbf{d}_l)})^2} (f'_\theta(\mathbf{d}_k) - f'_\theta(\mathbf{d}_l)).
\end{aligned}
\tag{3.4}
$$

The $\rho$ function is based on the reversed pair ranking $R^*(d_k, d_l, R)$, which is the same ranking as $R$ with the position of $d_k$ and $d_l$ swapped. An example of a reversed pair ranking is illustrated in Figure 3.1b. The idea is that if a preference for $d_k >_{\mathbf{c}} d_l$ is inferred in $R$ and both documents are equally relevant, then the reverse preference $d_l >_{\mathbf{c}} d_k$ is equally likely to be inferred in $R^*(d_k, d_l, R)$. The $\rho$ function reweighs the found preferences to the ratio between the probabilities of $R$ or $R^*(d_k, d_l, R)$ occurring:

$$
\rho(d_k, d_l, R, D) = \frac{P(R^*(d_k, d_l, R)|D)}{P(R|D) + P(R^*(d_k, d_l, R)|D)}.
\tag{3.5}
$$

This procedure has similarities with importance sampling [93]; however, we found that reweighing according to the ratio between $R$ and $R^*$ provides a more stable performance, since it produces less extreme values. Section 3.3.2 details exactly how $\rho$ creates an unbiased gradient.

    Algorithm 3.1 describes the PDGD method step by step: Given the initial parameters $\theta_1$ and a differentiable scoring function $f$ (Line 1), the method waits for a user-issued query $q_t$ to arrive (Line 3). Then the preselected set of documents $D_t$ for the query is fetched (Line 4), in our experiments these preselections are given in the LTR datasets that we use. A result list $R$ is sampled from the current model (Line 5 and Equation 3.1) and displayed to the user. The clicks from the user are logged (Line 6) and preferences between the displayed documents inferred (Line 8). The gradient is initialized (Line 7), and for each pair document pair $d_k$, $d_l$ such that $d_k >_{\mathbf{c}} d_l$, the weight $\rho(d_k, d_l, R, D)$ is calculated (Line 9 and Equation 3.5), followed by the gradient for the pair probability

(Line 10 and Equation 3.4). Finally, the gradient for the scoring function $f$ is weighted and added to the gradient (Line 11), resulting in the estimated gradient. The model is then updated by taking an $\eta$ step in the direction of the gradient (Line 12). The algorithm again waits for the next query to arrive and thus the process continues indefinitely.

PDGD has some notable advantages over MGD [111]. Firstly, it explicitly models uncertainty over the documents per query, thus PDGD is able to have high confidence in its ranking for one query, while being completely uncertain for another query. As a result, it will vary the amount of exploration per query, allowing it to avoid exploration in cases where it is not required and focussing on areas where it can improve. In contrast, MGD does not explicitly model confidence: its degree of exploration is only affected by the norm of its linear model [80]. Consequently, MGD is unable to vary exploration per query nor is there a way to directly measure its level of confidence. Secondly, PDGD works for any differentiable scoring function $f$ and does not rely on sampling model variants. Conversely, MGD is based around sampling from the unit sphere around a model; this approach is very ineffective for non-linear models. Additionally, sampling large models and producing rankings for them can be very computationally expensive. Besides these beneficial properties, our experimental results in Section 3.5 show that PDGD achieves significantly higher levels of performance than MGD and other previous methods.

## 3.3.2 Unbiased gradient estimation

The previous section introduced PDGD; this section answers **RQ3.2**:

**RQ3.2** Is the gradient estimation of PDGD unbiased?

First, Theorem 3.1 will provide a definition of unbiasedness w.r.t. user document pair preferences. Then we state the assumptions we make about user behavior and use them to prove Theorem 3.1. Our notation will use $d_k =_{rel} d_l$ to indicate no user preference between two documents t $d_k$ and $d_l$; and $d_k >_{rel} d_l$ to indicate a preference for $d_k$ over $d_l$; and $d_k <_{rel} d_l$ for the opposite preference.

**Theorem 3.1.** *The expected estimated gradient of PDGD can be written as a weighted sum, with a unique weight $\alpha_{k,l}$ for each possible document pair $d_k$ and $d_l$ in the document collection $D$:*

$$E[\nabla f_\theta(\cdot)] = \sum_{d_k, d_l \in D} \alpha_{k,l}(f'_{\theta_t}(\mathbf{d}_k) - f'_{\theta_t}(\mathbf{d}_l)). \tag{3.6}$$

*The signs of the weights $\alpha_{k,l}$ adhere to user preferences between documents. That is, if there is no preference:*

$$d_k =_{rel} d_l \Leftrightarrow \alpha_{k,l} = 0; \tag{3.7}$$

*if $d_k$ is preferred over $d_l$:*

$$d_k >_{rel} d_l \Leftrightarrow \alpha_{k,l} > 0; \tag{3.8}$$

*and if $d_l$ is preferred over $d_k$:*

$$d_k <_{rel} d_l \Leftrightarrow \alpha_{k,l} < 0. \tag{3.9}$$

*Therefore, in expectation PDGD will perform updates that adhere to the preferences between the documents in every possible document pair.*

**Assumptions.** To prove Theorem 3.1 the following assumptions about user behavior will be used:

**Assumption 1.** We assume that clicks from a user are position biased and conditioned on the relevance of the current document and the previously considered documents. For a click on a document in ranking $R$ at position $i$ the probability can be written as:

$$P(click(R_i)|\{R_0, \ldots, R_{i-1}, R_{i+1}\}). \quad (3.10)$$

For ease of notation, we will denote the set of "other documents" as $\{\ldots\}$ from here on.

**Assumption 2.** If there is no user preference between two documents $d_k, d_l$, denoted by $d_k =_{rel} d_l$, we assume that each is equally likely to be clicked given the same context:

$$d_k =_{rel} d_l \Rightarrow P(click(d_k)|\{\ldots\}) = P(click(d_l)|\{\ldots\}). \quad (3.11)$$

**Assumption 3.** If a document in the set of documents being considered is replaced with an equally preferred document the click probability is not affected:

$$d_k =_{rel} d_l \Rightarrow P(click(R_i)|\{\ldots, d_k\}) = P(click(R_i)|\{\ldots, d_l\}). \quad (3.12)$$

**Assumption 4.** Similarly, given the same context if one document is preferred over another, then it is more likely to be clicked:

$$d_k >_{rel} d_l \Rightarrow P(click(d_k)|\{\ldots\}) > P(click(d_l)|\{\ldots\}). \quad (3.13)$$

**Assumption 5.** Lastly, for any pair $d_k >_{rel} d_l$, the considered document set $\{\ldots, d_k\}$ and the same set with $d_k$ replaced by $d_l$ $\{\ldots, d_l\}$, we assume that the preferred $d_k$ in the context of $\{\ldots, d_l\}$ is more likely to be clicked than $d_l$ in the context of $\{\ldots, d_k\}$:

$$d_k >_{rel} d_l \Rightarrow P(click(d_k)|\{\ldots, d_k\}) > P(click(d_l)|\{\ldots, d_l\}). \quad (3.14)$$

These are all the assumptions we make about the user. With these assumptions, we can proceed to prove Theorem 3.1.

*Proof of Theorem 3.1.* We denote the probability of inferring the preference of $d_k$ over $d_l$ in ranking $R$ as $P(d_k >_{\mathbf{c}} d_l|R)$. Then the expected gradient $\nabla f_\theta(\cdot)$ of PDGD can be written as:

$$E[\nabla f_\theta(\cdot)] =$$
$$\sum_R \sum_{d_k, d_l \in D} \left[ P(d_k >_{\mathbf{c}} d_l|R) \cdot P(R) \cdot \rho(d_k, d_l, R, D)[\nabla P(d_k \succ d_l)] \right]. \quad (3.15)$$

We will rewrite this expectation using the symmetry property of the reversed pair ranking:

$$R^n = R^*(d_k, d_l, R^m) \Leftrightarrow R^m = R^*(d_k, d_l, R^n). \quad (3.16)$$

First, we define a weight $\omega_{k,l}^R$ for every document pair $d_k, d_l$ and ranking $R$ so that:

$$\omega_{k,l}^R = P(R)\rho(d_k, d_l, R, D) = \frac{P(R|D)P(R^*(d_k, d_l, R)|D)}{P(R|D) + P(R^*(d_k, d_l, R)|D)}. \qquad (3.17)$$

Therefore, the weight for the reversed pair ranking is equal:

$$\begin{aligned} \omega_{k,l}^{R^*(d_k,d_l,R)} &= P(R^*(d_k, d_l, R))\rho(d_k, d_l, R^*(d_k, d_l, R), D) \\ &= \omega_{k,l}^R. \end{aligned} \qquad (3.18)$$

Then, using the symmetry of Equation 3.3 we see that:

$$\nabla P(d_k \succ d_l) = -\nabla P(d_l \succ d_k). \qquad (3.19)$$

Thus, with $R^*$ as a shorthand for $R^*(d_k, d_l, R)$, the expectation can be rewritten as:

$$E[\nabla f_\theta(\cdot)] = \qquad (3.20)$$
$$\sum_{d_k,d_l \in D} \sum_R \frac{\omega_{i,j}^R}{2} \left( P(d_k >_\mathbf{c} d_l \mid R) - P(d_l >_\mathbf{c} d_k \mid R^*) \right) \left[ \nabla P(d_k \succ d_l) \right],$$

proving that the expected gradient matches the form of Equation 3.6. Then to prove that Equations 3.7, 3.8, and 3.9 are correct we will show that:

$$d_k =_{rel} d_l \Rightarrow P(d_k >_\mathbf{c} d_l|R) = P(d_l >_\mathbf{c} d_k|R^*), \qquad (3.21)$$
$$d_k >_{rel} d_l \Rightarrow P(d_k >_\mathbf{c} d_l|R) > P(d_l >_\mathbf{c} d_k|R^*), \qquad (3.22)$$
$$d_k <_{rel} d_l \Rightarrow P(d_k >_\mathbf{c} d_l|R) < P(d_l >_\mathbf{c} d_k|R^*). \qquad (3.23)$$

If a preference $R_i >_\mathbf{c} R_j$ is inferred then there are only three possible cases based on the positions:

1. The clicked document succeeds the unclicked document by more than one position: $i + 1 > j$.

2. The clicked document precedes the unclicked document by more than one position: $i - 1 < j$.

3. The clicked document is one position before or after the unclicked document: $i = j + 1 \vee i = j - 1$.

In the first case the clicked document succeeds the other by more than one position, the probability of an inferred preference is then:

$$i + 1 > j \Rightarrow P(R_i >_\mathbf{c} R_j|R) = P(\mathbf{c}_i|R_i, \{\ldots, R_j\})(1 - P(\mathbf{c}_j|R_j, \{\ldots\})). \qquad (3.24)$$

Combining Assumption 2 and 3 with Equation 3.24 proves Equation 3.21 for this case. Furthermore, combining Assumption 4 and 5 with Equation 3.24 proves Equations 3.22 and 3.23 for this case as well.

Table 3.1: Instantiations of Cascading Click Models [36] as used for simulating user behavior in experiments.

| $R$ | $P(click = 1 \mid R)$ | | | | | $P(stop = 1 \mid click = 1, R)$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | 4 |
| *perfect* | 0.0 | 0.2 | 0.4 | 0.8 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| *navigational* | 0.05 | 0.3 | 0.5 | 0.7 | 0.95 | 0.2 | 0.3 | 0.5 | 0.7 | 0.9 |
| *informational* | 0.4 | 0.6 | 0.7 | 0.8 | 0.9 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |

Then the second case is when the clicked document appears more than one position before the unclicked document, the probability of the inferred preference is then:

$$i + 1 < j \Rightarrow \\ P(R_i >_\mathbf{c} R_j | R) = P(\mathbf{c}_i | R_i, \{\ldots\})(1 - P(\mathbf{c}_j | R_j, \{\ldots, R_i\}))P(\mathbf{c}_{rem}), \tag{3.25}$$

where $P(\mathbf{c}_{rem})$ denotes the probability of an additional click that is required to add $R_j$ to the inferred observed documents. First, due to Assumption 1 this probability will be the same for $R$ and $R^*$:

$$P(\mathbf{c}_{rem} | R_i, R_j, R) = P(\mathbf{c}_{rem} | R_i, R_j, R^*). \tag{3.26}$$

Combining Assumption 2 and 3 with Equation 3.25 also proves Equation 3.21 for this case. Furthermore, combining Assumption 4 and 5 with Equation 3.25 also proves Equation 3.22 and 3.23 for this case as well.

Lastly, in the third case the clicked document is one position before or after the other document, the probability of the inferred preference is then:

$$i = j + 1 \lor i = j - 1 \tag{3.27} \\ \Rightarrow P(R_i >_\mathbf{c} R_j | R) = P(\mathbf{c}_i | R_i, \{\ldots, R_j\})(1 - P(\mathbf{c}_j | R_j, \{\ldots, R_i\})).$$

Combining Assumption 3 with Equation 3.28 proves Equation 3.21 for this case as well. Then, combining Assumption 5 with Equation 3.28 also proves Equation 3.22 and 3.23 for this case. □

This concludes our proof of the unbiasedness of PDGD. Hence, we answer **RQ3.2** positively: the gradient estimation of PDGD is unbiased. We have shown that the expected gradient is in line with user preferences between document pairs.

## 3.4   Experiments

In this section we detail the experiments that were performed to answer the research questions in Section 3.1.

### 3.4.1   Datasets

Our experiments are performed over five publicly available LTR datasets; we have selected three large labelled dataset from commercial search engines and two smaller research datasets. Every dataset consists of a set of queries with each query having a corresponding preselected document set. The exact content of the queries and documents are unknown, each query is represented only by an identifier, but each query-document pair has a feature representation and relevance label. Depending on the dataset, the relevance labels are graded differently; we have purposefully chosen datasets that have at least two grades of relevance. Each dataset is divided in training, validation and test partitions.

The oldest datasets we use are *MQ2007* and *MQ2008* [95] which are based on the Million Query Track [8] and consist of 1,700 and 800 queries. They use representations of 46 features that encode ranking models such as TF.IDF, BM25, Language Modeling, PageRank, and HITS on different parts of the documents. They are divided into five folds and the labels are on a three-grade scale from *not relevant* (0) to *very relevant* (2).

In 2010 Microsoft released the *MSLR-WEB30k* and *MLSR-WEB10K* datasets [95], which are both created from a retired labelling set of a commercial web search engine (Bing). The former contains 30,000 queries with each query having 125 assessed documents on average, query-document pairs are encoded in 136 features, The latter is a subsampling of 10,000 queries from the former dataset. For practical reasons only *MLSR-WEB10K* was used for this chapter. Also in 2010 Yahoo! released an LTR dataset [17]. It consists of 29,921 queries and 709,877 documents encoded in 700 features, all sampled from query logs of the Yahoo! search engine. Finally, in 2016 a LTR dataset was released by the Istella search engine [27]. It is the largest with 33,118 queries, an average of 315 documents per query and 220 features. These three commercial datasets all label relevance on a five-grade scale: from *not relevant* (0) to *perfectly relevant* (4).

### 3.4.2   Simulating user behavior

For simulating users we follow the standard setup for OLTR simulations [38, 40, 90, 111, 137]. First, queries issued by users are simulated by uniformly sampling from the static dataset. Then the algorithm determines the result list of documents to display. User interactions with the displayed list are then simulated using a *cascade click model* [20, 36]. This models a user who goes through the documents one at a time in the displayed order. At each document, the user decides whether to click it or not, modelled as a probability conditioned on the relevance label $R$: $P(click = 1 \mid R)$. After a click has occurred, the user's information need may be satisfied and they may then stop considering documents. The probability of a user stopping after a click is modelled as $P(stop = 1 \mid click = 1, R)$. For our experiments $\kappa = 10$ documents are displayed at each impression.

The three instantiations of cascade click models that we used are listed in Table 3.1. First, a *perfect* user is modelled who considers every document and solely clicks on all relevant documents. The second models a user with a *navigational* task, where a single highly relevant document is searched. Finally, an *informational* instantiation models a user without a specific information need, and thus typically clicks on many documents.

These models have varying levels of noise, as each behavior depends on the relevance labels of documents with a different degree.

### 3.4.3 Experimental runs

For our experiments three baselines are used. First, MGD with Probabilistic Multi-leaving [90]; this is the highest performing existing OLTR method [80, 90]. For this chapter $n = 49$ candidates were sampled per iteration from the unit sphere with $\delta = 1$; updates are performed with $\eta = 0.01$ and zero initialization was used. Additionally, DBGD is used for comparison as it is one of the most influential methods, it was run with the same parameters except that only $n = 1$ candidate is sampled per iteration. Furthermore, we also let DBGD optimize a single hidden-layer neural network with 64 hidden nodes and sigmoid activation functions with *Xavier* initialization [33]. These parameters were also found most effective in previous work [40, 90, 111, 132].

Additionally, the pairwise method introduced by Hofmann et al. [40] is used as a baseline. Despite not showing significant improvements over DBGD in the past [40], the comparison with PDGD is interesting because they both estimate gradients from pairwise preferences. For this baseline, $\eta = 0.01$ and $\epsilon = 0.8$ is used; these parameters are chosen to maximize the performance at convergence [40].

Runs with PDGD are performed with both a linear and neural ranking model. For the linear ranking model $\eta = 0.1$ and zero initialization was used. The neural network has the same parameters as the one optimized by DBGD, except for $\eta = 0.1$.

### 3.4.4 Metrics and tests

Two aspects of performance are evaluated seperately: the final convergence and the ranking quality during training.

Final convergence is addressed in *offline performance* which is the average NDCG@10 of the ranking model over the queries in the held-out test-set. The offline performance is measured after 10,000 impressions at which point most ranking models have reached convergence. The user experience during optimization should be considered as well, since deterring users during training would compromise the goal of OLTR. To address this aspect of evaluation *online performance* has been introduced [39]; it is the cumulative discounted NDCG@10 of the rankings displayed during training. For $T$ sequential queries with $R^t$ as the ranking displayed to the user at timestep $t$, this is:

$$Online\_Performance = \sum_{t=1}^{T} NDCG(R^t) \cdot \gamma^{(t-1)}. \qquad (3.28)$$

This metric models the expected reward a user receives with a $\gamma$ probability that the user stops searching after each query. We follow previous work [80, 90] by choosing a discount factor of $\gamma = 0.9995$, consequently queries beyond the horizon of 10,000 queries have a less than $1\%$ impact.

Lastly, all experimental runs are repeated 125 times, spread evenly over the available dataset folds. Results are averaged and a two-tailed Student's t-test is used for significance testing. In total, our results are based on more than 90,000,000 user impressions.

## 3.5   Results and Analysis

Our main results are displayed in Table 3.2 and Table 3.3, showing the offline and online performance of all methods, respectively. Additionally, Figure 3.2 displays the offline performance on the MSLR-WEB10k dataset over 30,000 impressions and Figure 3.3 over 1,000,000 impressions. We use these results to answer **RQ3.1** – whether PDGD provides significant improvements over existing OLTR methods – and **RQ3.3** – whether PDGD is successful at optimizing different types of ranking models.

### 3.5.1   Convergence of ranking models

First, we consider the offline performance after 10,000 impressions as reported in Table 3.2. We see that the DBGD and MGD baselines reach similar levels of performance, with marginal differences at low levels of noise. Our results seem to suggest that MGD provides an efficient alternative to DBGD that requires fewer user interactions and is more robust to noise. However, MGD does not appear to have an improved point of convergence over DBGD, Figure 3.2 further confirms this conclusion. Additionally, Table 3.2 and Figure 3.3 reveal thats DBGD is incapable of training its neural network so that it improves over the linear model, even after 1,000,000 impressions.

Alternatively, the pairwise baseline displays different behavior, providing improvements over DBGD and MGD on most datasets under all levels of noise. However, on the istella dataset large decreases in performance are observed. Thus it is unclear if this method provides a reliable alternative to DBGD or MGD in terms of convergence. Figure 3.2 also reveals that it converges within several hundred impressions, while DBGD or MGD continue to learn and considerably improve over the total 30,000 impressions. Because the pairwise baseline also converges sub-optimally under the perfect click model, we do not attribute its suboptimal convergence to noise but to the method being biased.

Conversely, Table 3.2 shows that PDGD reaches significantly higher performance than all the baselines within 10,000 impressions. Improvements are observed on all datasets under all levels of noise, especially on the commercial datasets where increases up to 0.17 NDCG are observed. Our results also show that PDGD learns faster than the baselines; at all time-steps the offline performance of PDGD is at least as good or better than all other methods, across all datasets. This increased learning speed can also be observed in Figure 3.2. Besides the faster learning it also appears as if PDGD converges at a better optimum than DBGD or MGD. However, Figure 3.2 reveals that DBGD does not fully converge within 30,000 iterations. Therefore, we performed an additional experiment where PDGD and DBGD optimize models over 1,000,000 impressions on the MSLR-WEB10k dataset, as displayed in Figure 3.3. Clearly the performance of DBGD plateaus at a considerably lower level than that of PDGD. Therefore, we conclude that PDGD indeed has an improved point of final convergence compared to DBGD and MGD.

Finally, Figure 3.2 and 3.3 also shows the behavior predicted by the speed-quality tradeoff [80]: a more complex model will have a worse initial performance but a better final convergence. Here, we see that depending on the level of interaction noise the neural model requires 3,000 to 20,000 iterations to match the performance of a linear

model. However, in the long run the neural model does converge at a significantly better point of convergence. Thus, we conclude that PDGD is capable of effectively optimizing different kinds of models in terms of offline performance.

In conclusion, our results show that PDGD learns faster than existing OLTR methods while also converging at significantly better levels of performance.

### 3.5.2    User experience during training

Besides the ranking models learned by the OLTR methods, we also consider the user experience during optimization. Table 3.3 shows that the online performance of DBGD and MGD are close to each other; MGD has a higher online performance due to its faster learning speed [90, 111]. In contrast, the pairwise baseline has a substantially lower online performance in all cases. Because Figure 3.2 shows that the learning speed of the pairwise baseline sometimes matches that of DBGD and MGD, we attribute this difference to the exploration strategy it uses. Namely, the random insertion of uniformly sampled documents by this baseline appears to have a strong negative effect on the user experience.

The linear model optimized by PDGD has significant improvements over all baseline methods on all datasets and under all click models. This improvement indicates that the exploration of PDGD, which uses a distribution over documents, does not lead to a worse user experience. In conclusion, PDGD provides a considerably better user experience than all existing methods.

Finally, we also discuss the performance of the neural models optimized by PDGD and DBGD. This model has both significant increases and decreases in online performance varying per dataset and amount of interaction noise. The decrease in user experience is predicted by the speed-quality tradeoff [80], as Figure 3.2 also shows, the neural model has a slower learning speed leading to a worse initial user experience. A solution to this tradeoff has been proposed by Oosterhuis and de Rijke [80], which optimizes a cascade of models. In this case, the cascade could combine the user experience of the linear model with the final convergence of the neural model, providing the best of both worlds.

### 3.5.3    Improvements of PDGD

After having discussed the offline and online performance of PDGD, we will now answer **RQ3.1** and **RQ3.3**.

First, concerning **RQ3.1** (whether PDGD performs significantly better than MGD), the results of our experiments show that models optimized with PDGD learn faster and converge at better optima than MGD, DBGD, and the pairwise baseline, regardless of dataset or level of interaction noise. Moreover, the level of performance reached with PDGD is significantly higher than the final convergence of any other method. Thus, even in the long run DBGD and MGD are incapable of reaching the offline performance of PDGD. Additionally, the online performance of a linear model optimized with PDGD is significantly better across all datasets and user models. Therefore, we answer **RQ3.1** positively: PDGD outperforms existing methods both in terms of model convergence and user experience during learning.

Then, with regards to **RQ3.3** (whether PDGD can effectively optimize different types of models), in our experiments we have successfully optimized models from two families: linear models and neural networks. Both models reach a significantly higher level of performance of model convergence than previous OLTR methods, across all datasets and degrees of interaction noise. As expected, the simpler linear model has a better initial user experience, while the more complex neural model has a better point of convergence. In conclusion, we answer **RQ3.3** positively: PDGD is applicable to different ranking models and effective for both linear and non-linear models.

## 3.6   Conclusion

In this chapter, we have introduced a novel OLTR method: PDGD that estimates its gradient using inferred pairwise document preferences. In contrast with previous OLTR approaches PDGD does not rely on online evaluation to update its model. Instead after each user interaction it infers preferences between document pairs. Subsequently, it constructs a pairwise gradient that updates the ranking model according to these preferences.

We have proven that this gradient is unbiased w.r.t. user preferences, that is, if there is a preference between a document pair, then in expectation the gradient will update the model to meet this preference. Furthermore, our experimental results show that PDGD learns faster and converges at a higher performance level than existing OLTR methods. Thus, it provides better performance in the short and long term, leading to an improved user experience during training as well. On top of that, PDGD is also applicable to any differentiable ranking model, in our experiments a linear and a neural network were optimized effectively. Both reached significant improvements over DBGD and MGD in performance at convergence. In conclusion, the novel unbiased PDGD algorithm provides better performance than existing methods in terms of convergence and user experience. Unlike the previous state-of-the-art, it can be applied to any differentiable ranking model.

We can now answer thesis research question **RQ2** positively: OLTR is possible without relying on model-sampling and online evaluation. Moreover, our results shows that using PDGD instead leads to much higher performance, and is much more effective at optimizing non-linear models.

Future research could consider the regret bounds of PDGD; these could give further insights into why it outperforms DBGD based methods. Furthermore, while we proved the unbiasedness of our method w.r.t. document pair preferences, the expected gradient weighs document pairs differently. Offline LTR methods like LambdaMART [13] use a weighted pairwise loss to create a listwise method that directly optimizes IR metrics. However, in the online setting there is no metric that is directly optimized. Instead, future work could see if different weighing approaches are more in line with user preferences. Another obvious avenue for future research is to explore the effectiveness of different ranking models in the online setting. There is a large collection of research in ranking models in offline LTR, with the introduction of PDGD such an extensive exploration in models is now also possible in OLTR.

Based on the big difference in observed performance between PDGD and DBGD,

Chapter 4 will further extend this comparison to more extreme experimental conditions. Furthermore, Chapter 8 will also consider the performance of PDGD and compare it with methods inspired by counterfactual LTR. Additionally, Chapter 8 will consider applying PDGD as a counterfactual method and without debiasing weights, and finds that in both these scenarios this leads to biased convergence.
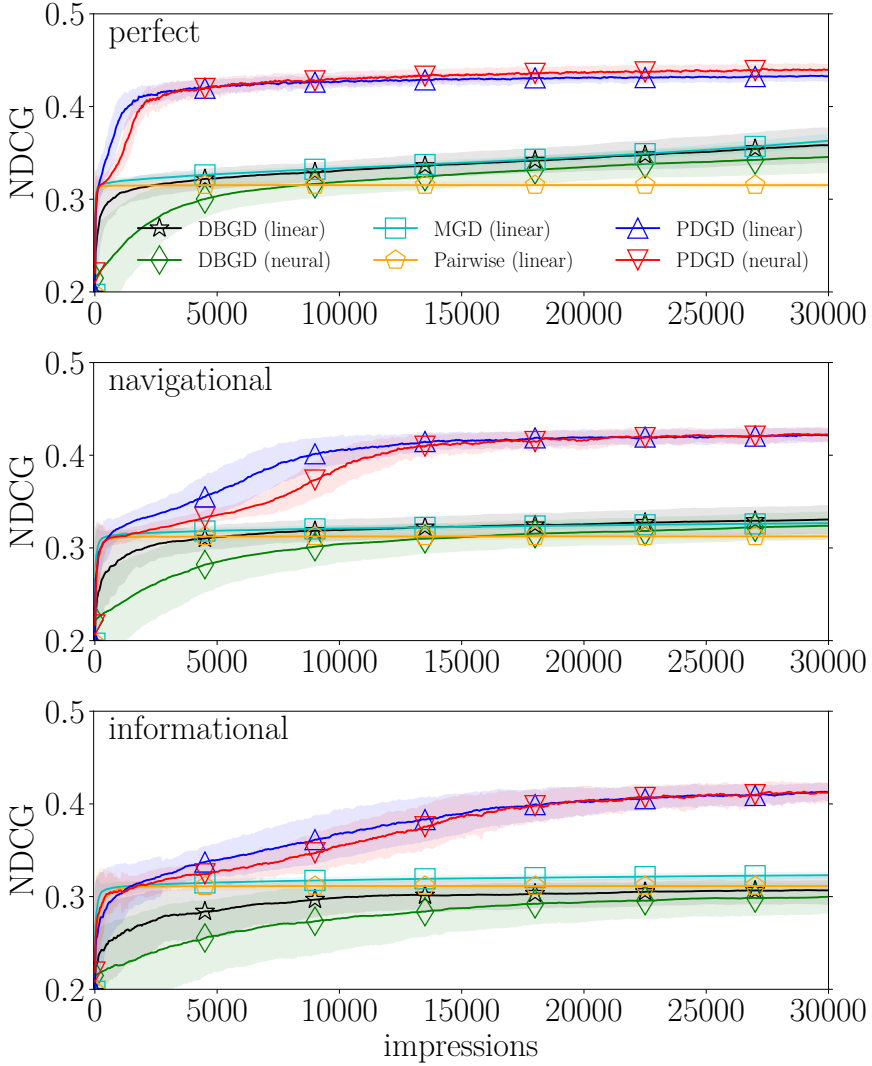
Figure 3.2: Offline performance (NDCG) on the MSLR-WEB10k dataset under three different click models, the shaded areas indicate the standard deviation.
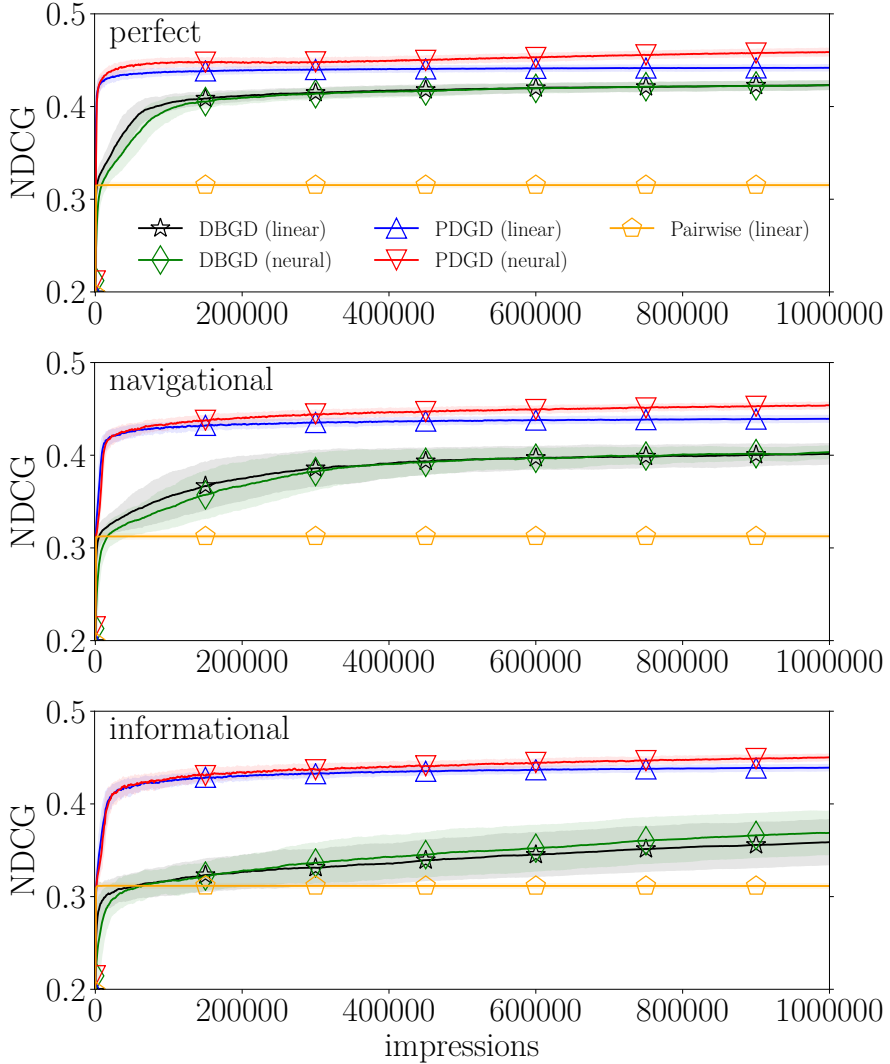
Figure 3.3: Long-term offline performance (NDCG) on the MSLR-WEB10k dataset under three click models, the shaded areas indicate the standard deviation.

Table 3.2: Offline performance (NDCG) for different instantiations of CCM (Table 3.1). The standard deviation is shown in brackets, bold values indicate the highest performance per dataset and click model, significant improvements over the DBGD, MGD and pairwise baselines are indicated by $^{\triangle}$ ($p < 0.05$) and $^{\blacktriangle}$ ($p < 0.01$), no losses were measured.

| | MQ2007 | MQ2008 | MSLR-WEB10k | Yahoo | istella |
|---|---|---|---|---|---|
| *perfect* | | | | | |
| DBGD (linear) | 0.483 (0.023) | 0.683 (0.024) | 0.331 (0.010) | 0.684 (0.010) | 0.448 (0.014) |
| DBGD (neural) | 0.463 (0.025) | 0.670 (0.026) | 0.319 (0.014) | 0.676 (0.016) | 0.429 (0.017) |
| MGD (linear) | 0.494 (0.022) | 0.690 (0.019) | 0.333 (0.003) | 0.714 (0.002) | 0.496 (0.004) |
| Pairwise (linear) | 0.479 (0.022) | 0.674 (0.017) | 0.315 (0.003) | 0.709 (0.001) | 0.252 (0.002) |
| PDGD (linear) | **0.511** (0.017) ▲ ▲ ▲ | **0.699** (0.024) ▲ ▲ ▲ | 0.427 (0.005) ▲ ▲ ▲ | **0.736** (0.004) ▲ ▲ ▲ | 0.573 (0.004) ▲ ▲ ▲ |
| PDGD (neural) | 0.509 (0.020) ▲ ▲ ▲ | 0.698 (0.024) ▲ ▲ ▲ | **0.430** (0.006) ▲ ▲ ▲ | 0.733 (0.005) ▲ ▲ ▲ | **0.575** (0.006) ▲ ▲ ▲ |
| *navigational* | | | | | |
| DBGD (linear) | 0.461 (0.025) | 0.670 (0.025) | 0.319 (0.011) | 0.661 (0.023) | 0.401 (0.015) |
| DBGD (neural) | 0.430 (0.033) | 0.646 (0.031) | 0.304 (0.019) | 0.649 (0.029) | 0.382 (0.024) |
| MGD (linear) | 0.426 (0.020) | 0.662 (0.015) | 0.321 (0.003) | 0.706 (0.009) | 0.405 (0.004) |
| Pairwise (linear) | 0.476 (0.022) | 0.677 (0.018) | 0.312 (0.003) | 0.696 (0.004) | 0.209 (0.002) |
| PDGD (linear) | **0.496** (0.019) ▲ ▲ ▲ | **0.695** (0.021) ▲ ▲ ▲ | **0.406** (0.015) ▲ ▲ ▲ | **0.725** (0.005) ▲ ▲ ▲ | **0.540** (0.008) ▲ ▲ ▲ |
| PDGD (neural) | 0.493 (0.020) ▲ ▲ ▲ | 0.692 (0.019) ▲ ▲ ▲ | 0.386 (0.006) ▲ ▲ ▲ | 0.722 (0.006) ▲ ▲ ▲ | 0.532 (0.011) ▲ ▲ ▲ |
| *informational* | | | | | |
| DBGD (linear) | 0.411 (0.036) | 0.631 (0.036) | 0.299 (0.017) | 0.620 (0.035) | 0.360 (0.028) |
| DBGD (neural) | 0.383 (0.047) | 0.595 (0.053) | 0.276 (0.033) | 0.603 (0.040) | 0.316 (0.057) |
| MGD (linear) | 0.406 (0.021) | 0.647 (0.036) | 0.318 (0.003) | 0.676 (0.043) | 0.387 (0.005) |
| Pairwise (linear) | 0.478 (0.022) | 0.677 (0.018) | 0.311 (0.003) | 0.690 (0.006) | 0.183 (0.001) |
| PDGD (linear) | **0.487** (0.021) ▲ ▲ ▲ | **0.690** (0.022) ▲ ▲ ▲ | **0.368** (0.025) ▲ ▲ ▲ | **0.713** (0.008) ▲ ▲ ▲ | **0.532** (0.010) ▲ ▲ ▲ |
| PDGD (neural) | 0.483 (0.022) ▲ ▲ ▲ | 0.686 (0.022) ▲ ▲ ▲ | 0.355 (0.021) ▲ ▲ ▲ | 0.709 (0.009) ▲ ▲ ▲ | 0.525 (0.012) ▲ ▲ ▲ |

Table 3.3: Online performance (Discounted Cumulative NDCG, Section 3.4.4) for different instantiations of CCM (Table 3.1). The standard deviation is shown in brackets, bold values indicate the highest performance per dataset and click model, significant improvements and losses over the DBGD, MGD and pairwise baselines are indicated by $^{\triangle}$ (p < 0.05) and $^{\blacktriangle}$ (p < 0.01) and by $^{\triangledown}$ and $^{\blacktriangledown}$, respectively.

| | MQ2007 | MQ2008 | MSLR-WEB10k | Yahoo | istella |
|---|---|---|---|---|---|
| | | | *perfect* | | |
| DBGD (linear) | 675.7 (21.8) | 843.6 (40.8) | 533.6 (15.6) | 1159.3 (31.6) | 589.9 (19.2) |
| DBGD (neural) | 602.7 (58.1) | 776.9 (67.4) | 481.2 (53.0) | 1135.7 (41.3) | 494.3 (60.5) |
| MGD (linear) | 689.6 (15.3) | 858.6 (40.6) | 558.7 (6.4) | 1203.9 (9.9) | 670.9 (8.6) |
| Pairwise (linear) | 458.4 (13.3) | 616.6 (25.8) | 345.3 (4.6) | 1027.2 (9.2) | 64.5 (2.1) |
| PDGD (linear) | **797.3** (17.3) ▲▲▲ | **959.7** (43.4) ▲▲▲▲ | **691.4** (12.3) ▲▲▲▲ | **1360.3** (10.8) ▲▲▲▲ | **957.5** (9.4) ▲▲▲▲ |
| PDGD (neural) | 743.7 (18.8) ▲▲▲▲ | 925.4 (43.3) ▲▲▲▲ | 619.2 (13.6) ▲▲▲▲ | 1319.6 (22.2) ▲▲▲▲ | 834.0 (22.2) ▲▲▲▲ |
| | | | *navigational* | | |
| DBGD (linear) | 638.6 (29.7) | 816.9 (42.0) | 508.2 (21.6) | 1129.9 (32.2) | 538.2 (29.0) |
| DBGD (neural) | 573.7 (68.4) | 740.3 (69.7) | 465.8 (52.0) | 1116.0 (45.7) | 414.3 (96.2) |
| MGD (linear) | 635.9 (14.7) | 824.5 (34.0) | 538.1 (7.6) | 1181.7 (20.0) | 593.2 (9.7) |
| Pairwise (linear) | 459.9 (12.9) | 618.6 (25.2) | 347.3 (5.4) | 1031.2 (9.0) | 72.6 (2.2) |
| PDGD (linear) | **703.0** (17.9) ▲▲▲▲ | **903.1** (40.7) ▲▲▲▲ | **578.1** (16.0) ▲▲▲▲ | **1298.4** (33.4) ▲▲▲▲ | **704.1** (33.5) ▲▲▲▲ |
| PDGD (neural) | 560.9 (14.6) ▼▽▲ | 788.7 (38.5) ▼▲▲ | 448.1 (12.3) ▼▼▲ | 1176.1 (17.0) ▲▽▲ | 390.2 (35.1) ▲▼▲ |
| | | | *informational* | | |
| DBGD (linear) | 584.2 (41.1) | 757.4 (56.9) | 477.2 (32.2) | 1110.0 (37.0) | 436.8 (57.4) |
| DBGD (neural) | 550.8 (75.7) | 720.9 (79.0) | 444.7 (60.9) | 1091.2 (48.6) | 322.9 (121.0) |
| MGD (linear) | 618.8 (21.7) | 815.1 (44.5) | 540.0 (7.7) | 1159.1 (40.0) | 581.8 (10.7) |
| Pairwise (linear) | 462.6 (14.4) | 619.6 (25.0) | 349.7 (6.6) | 1034.1 (9.0) | 77.0 (2.4) |
| PDGD (linear) | **704.8** (30.5) ▲▲▲▲ | **907.9** (42.0) ▲▲▲ | **567.3** (36.5) ▲▽▲▲ | **1266.7** (50.0) ▲▲▲▲ | **731.5** (80.0) ▲▲▲▲ |
| PDGD (neural) | 594.6 (23.0) △▲▼▲ | 818.3 (39.6) ▲▲▲ | 470.1 (19.4) ▽▲▼▲ | 1178.1 (22.8) ▲▲▲ | 484.3 (64.8) ▲▲▼▲ |

## 3.A   Notation Reference for Chapter 3

| Notation | Description |
| --- | --- |
| $q$ | a user-issued query |
| $d$, $d_k$, $d_l$ | document |
| $\mathbf{d}$ | feature representation of a query-document pair |
| $D$ | set of documents |
| $R$ | ranked list |
| $R^*$ | the reversed pair ranking $R^*(d_k, d_l, R)$ |
| $R_i$ | document placed at rank $i$ |
| $\rho$ | preference pair weighting function |
| $\theta$ | parameters of the ranking model |
| $f_\theta(\cdot)$ | ranking model with parameters $\theta$ |
| $f(\mathbf{d}_k)$ | ranking score for a document from model |
| $click(d)$ | a click on document $d$ |
| $d_k =_{rel} d_l$ | two documents equally preferred by users |
| $d_k >_{rel} d_l$ | a user preference between two documents |
| $d_k >_{\mathbf{c}} d_l$ | document preference inferred from clicks |

# 4

# A Critical Comparison of Online Learning to Rank Methods

Online Learning to Rank (OLTR) methods optimize ranking models by directly interacting with users, which allows them to be very efficient and responsive. All OLTR methods introduced during the past decade have extended on the original OLTR method: Dueling Bandit Gradient Descent (DBGD). In Chapter 3, a fundamentally different approach was introduced with the Pairwise Differentiable Gradient Descent (PDGD) algorithm. The empirical comparisons in Chapter 3 suggested that PDGD converges at much higher levels of performance and learns considerably faster than DBGD-based methods. In contrast, DBGD appeared unable to converge on the optimal model in scenarios with little noise or bias. Furthermore, it seemed DBGD is not effective at optimizing non-linear models. These observations are quite surprising and prompted us to further investigate DBGD. As a result, this Chapter will address the thesis research question:

**RQ3** *Are DBGD Learning to Rank methods reliable in terms of theoretical soundness and empirical performance?*

In this chapter, we investigate whether the previous conclusions about the PDGD and DBGD comparison generalize from ideal to worst-case circumstances. We do so in two ways. First, we compare the theoretical properties of PDGD and DBGD, by taking a critical look at previously proven properties in the context of ranking. Second, we estimate an upper and lower bound on the performance of methods by simulating both *ideal* user behavior and extremely *difficult* behavior, i.e., almost-random non-cascading user models. Our findings show that the theoretical bounds of DBGD do not apply to any common ranking model and, furthermore, that the performance of DBGD is substantially worse than PDGD in both ideal and worst-case circumstances. These results reproduce previously published findings about the relative performance of PDGD vs. DBGD and generalize them to extremely noisy and non-cascading circumstances. Overall they show that DBGD is a very flawed method for OLTR both in terms of theoretical guarantees and performance.

---

This chapter was published as [84]. Appendix 4.A gives a reference for the notation used in this chapter.

## 4.1   Introduction

Learning to Rank (LTR) plays a vital role in information retrieval. It allows us to optimize models that combine hundreds of signals to produce rankings, thereby making large collections of documents accessible to users through effective search and recommendation. Traditionally, LTR has been approached as a supervised learning problem, where annotated datasets provide human judgements indicating relevance. Over the years, many limitations of such datasets have become apparent: they are costly to produce [17, 95] and actual users often disagree with the relevance annotations [104]. As an alternative, research into LTR approaches that learn from user behavior has increased. By learning from the implicit feedback in user behavior, users' true preferences can potentially be learned. However, such methods must deal with the noise and biases that are abundant in user interactions [134]. Roughly speaking, there are two approaches to LTR from user interactions: learning from historical interactions and Online Learning to Rank (OLTR). Learning from historical data allows for optimization without gathering new data [58], though it does require good models of the biases in logged user interactions [20]. In contrast, OLTR methods learn by interacting with the user, thus they gather their own learning data. As a result, these methods can adapt instantly and are potentially much more responsive than methods that use historical data.

Dueling Bandit Gradient Descent (DBGD) [132] is the most prevalent OLTR method; it has served as the basis of the field for the past decade. DBGD samples variants of its ranking model, and compares them using interleaving to find improvements [41, 96]. Subsequent work in OLTR has extended on this approach [43, 111, 125]. In Chapter 3, the first alternative approach to DBGD was introduced with *Pairwise Differentiable Gradient Descent* (PDGD) [82]. PDGD estimates a pairwise gradient that is reweighed to be unbiased w.r.t. users' document pair preferences. Chapter 3 showed considerable improvements over DBGD under simulated user behavior [84]: a substantially higher point of performance at convergence and a much faster learning speed. The results in Chapter 3 are based on simulations using low-noise cascading click models. The pairwise assumption that PDGD makes, namely, that all documents preceding a clicked document were observed by the user, is always correct in these circumstances, thus potentially giving it an unfair advantage over DBGD. Furthermore, the low level of noise presents a close-to-ideal situation, and it is unclear whether the findings in Chapter 3 generalize to less perfect circumstances.

In this chapter, we contrast PDGD and DBGD. Prior to an experimental comparison, we determine whether there is a theoretical advantage of DBGD over PDGD and examine the regret bounds of DBGD for ranking problems. We then investigate whether the benefits of PDGD over DBGD reported in Chapter 3 generalize to circumstances ranging from ideal to worst-case. We simulate circumstances that are perfect for both methods – behavior without noise or position-bias – and circumstances that are the worst possible scenario – almost-random, extremely-biased, non-cascading behavior. These settings provide estimates of upper and lower bounds on performance, and indicate how well previous comparisons generalize to different circumstances. Additionally, we introduce a version of DBGD that is provided with an oracle interleaving method; its performance shows us the maximum performance DBGD could reach from hypothetical extensions.

In summary, we map thesis research question **RQ3** into the following more fine-grained research questions:

**RQ4.1** Do the regret bounds of DBGD provide a benefit over PDGD?

**RQ4.2** Do the advantages of PDGD over DBGD observed in Chapter 3 generalize to extreme levels of noise and bias?

**RQ4.3** Is the performance of PDGD reproducible under non-cascading user behavior?

## 4.2   Related Work

This section provides a brief overview of traditional LTR (Section 4.2.1), of LTR from historical interactions (Section 4.2.2), and OLTR (Section 4.2.3).

### 4.2.1   Learning to rank from annotated datasets

Traditionally, LTR has been approached as a supervised problem; in the context of OLTR this approach is often referred to as *offline* LTR. It requires a dataset containing relevance annotations of query-document pairs, after which a variety of methods can be applied [75]. The limitations of offline LTR mainly come from obtaining such annotations. The costs of gathering annotations are high as it is both time-consuming and expensive [17, 95]. Furthermore, annotators cannot judge for very specific users, i.e., gathering data for personalization problems is infeasible. Moreover, for certain applications it would be unethical to annotate items, e.g., for search in personal emails or documents [127]. Additionally, annotations are stationary and cannot account for (perceived) relevance changes [1, 71, 120]. Most importantly, though, annotations are not necessarily aligned with user preferences; judges often interpret queries differently from actual users [104]. As a result, there has been a shift of interest towards LTR approaches that do not require annotated data.

### 4.2.2   Learning to rank from historical interactions

The idea of LTR from user interactions is long-established; one of the earliest examples is the original pairwise LTR approach [54]. This approach uses historical click-through interactions from a search engine and considers clicks as indications of relevance. Though very influential and quite effective, this approach ignores the *noise* and *biases* inherent in user interactions. Noise, i.e., any user interaction that does not reflect the user's true preference, occurs frequently, since many clicks happen for unexpected reasons [104]. Biases are systematic forms of noise that occur due to factors other than relevance. For instance, interactions will only involve displayed documents, resulting in selection bias [127]. Another important form of bias in LTR is position bias, which occurs because users are less likely to consider documents that are ranked lower [134]. Thus, to effectively learn true preferences from user interactions, a LTR method should be robust to noise and handle biases correctly.

In recent years counterfactual LTR methods have been introduced that correct for some of the bias in user interactions. Such methods use inverse propensity scoring to

account for the probability that a user observed a ranking position [58]. Thus, clicks on positions that are observed less often due to position bias will have greater weight to account for that difference. However, the position bias must be learned and estimated somewhat accurately [5]. On the other side of the spectrum are click models, which attempt to model user behavior completely [20]. By predicting behavior accurately, the effect of relevance on user behavior can also be estimated [11, 127].

An advantage of these approaches over OLTR is that they only require historical data and thus no new data has to be gathered. However, unlike OLTR, they do require a fairly accurate user model, and thus they cannot be applied in cold-start situations.

### 4.2.3   Online learning to rank

OLTR differs from the approaches listed above because its methods intervene in the search experience. They have control over what results are displayed, and can learn from their interactions instantly. Thus, the online approach performs LTR by interacting with users directly [132]. Similar to LTR methods that learn from historical interaction data, OLTR methods have the potential to learn the true user preferences. However, they also have to deal with the noise and biases that come with user interactions. Another advantage of OLTR is that the methods are very responsive, as they can apply their learned behavior instantly. Conversely, this also brings a danger as an online method that learns incorrect preferences can also worsen the experience immediately. Thus, it is important that OLTR methods are able to learn reliably in spite of noise and biases. Thus, OLTR methods have a two-fold task: they have to simultaneously present rankings that provide a good user experience *and* learn from user interactions with the presented rankings.

The original OLTR method is Dueling Bandit Gradient Descent (DBGD); it approaches optimization as a dueling bandit problem [132]. This approach requires an online comparison method that can compare two rankers w.r.t. user preferences; traditionally, DBGD methods use interleaving. Interleaving methods take the rankings produced by two rankers and combine them in a single result list, which is then displayed to users. From a large number of clicks on the presented list the interleaving methods can reliably infer a preference between the two rankers [41, 96]. At each timestep, DBGD samples a candidate model, i.e., a slight variation of its current model, and compares the current and candidate models using interleaving. If a preference for the candidate is inferred, the current model is updated towards the candidate slightly. By doing so, DBGD will update its model continuously and should oscillate towards an inferred optimum. Section 4.3 provides a complete description of the DBGD algorithm.

Virtually all work in OLTR in the decade since the introduction of DBGD has used DBGD as a basis. A straightforward extension comes in the form of Multileave Gradient Descent [111] which compares a large number of candidates per interaction [81, 108, 109]. This leads to a much faster learning process, though in the long run this method does not seem to improve the point of convergence.

One of the earliest extensions of DBGD proposed a method for reusing historical interactions to guide exploration for faster learning [43]. While the initial results showed great improvements [43], later work showed performance drastically decreasing in the long term due to bias introduced by the historical data [90]. Unfortunately, OLTR work

---

**Algorithm 4.1** Dueling Bandit Gradient Descent (DBGD).

---

1: **Input**: initial weights: $\theta_1$; unit: $u$; learning rate $\eta$.
2: **for** $t \leftarrow 1 \ldots \infty$ **do**
3:     $q_t \leftarrow$ receive_query$(t)$              *// obtain a query from a user*
4:     $\theta_t^c \leftarrow \theta_t +$ sample_from_unit_sphere$(u)$     *// create candidate ranker*
5:     $R_t \leftarrow$ get_ranking$(\theta_t, D_{q_t})$       *// get current ranker ranking*
6:     $R_t^c \leftarrow$ get_ranking$(\theta_t^c, D_{q_t})$     *// get candidate ranker ranking*
7:     $I_t \leftarrow$ interleave$(R_t, R_t^c)$         *// interleave both rankings*
8:     $\mathbf{c}_t \leftarrow$ display_to_user$(I_t)$     *// displayed interleaved list, record clicks*
9:     **if** preference_for_candidate$(I_t, \mathbf{c}_t, R_t, R_t^c)$ **then**
10:         $\theta_{t+1} \leftarrow \theta_t + \eta(\theta_t^c - \theta_t)$     *// update model towards candidate*
11:     **else**
12:         $\theta_{t+1} \leftarrow \theta_t$            *// no update*

---

that continued this historical approach [125] also only considered short term results; moreover, the results of some work [135] are not based on held-out data. As a result, we do not know whether these extensions provide decent long-term performance and it is unclear whether the findings of these studies generalize to more realistic settings.

In Chapter 3, an inherently different approach to OLTR was introduced with PDGD [82]. PDGD interprets its ranking model as a distribution over documents; it estimates a pairwise gradient from user interactions with sampled rankings. This gradient is differentiable, allowing for non-linear models like neural networks to be optimized, something DBGD is ineffective at [80, 82]. Section 4.4 provides a detailed description of PDGD. In the chapter in which we introduced PDGD (Chapter 3), we claim that it provides substantial improvements over DBGD. However, those claims are based on cascading click models with low levels of noise. This is problematic because PDGD assumes a cascading user, and could thus have an unfair advantage in this setting. Furthermore, it is unclear whether DBGD with a perfect interleaving method could still improve over PDGD. Lastly, DBGD has proven regret bounds while PDGD has no such guarantees.

In this chapter, we clear up these questions about the relative strengths of DBGD and PDGD by comparing the two methods under non-cascading, high-noise click models. Additionally, by providing DBGD with an oracle comparison method, its hypothetical maximum performance can be measured; thus, we can study whether an improvement over PDGD is hypothetically possible. Finally, a brief analysis of the theoretical regret bounds of DBGD shows that they do not apply to any common ranking model, therefore hardly providing a guaranteed advantage over PDGD.

## 4.3 Dueling Bandit Gradient Descent

This section describes the DBGD algorithm in detail, before discussing the regret bounds of the algorithm.

### 4.3.1 The Dueling Bandit Gradient Descent method

The DBGD algorithm [132] describes an indefinite loop that aims to improve a ranking model at each step; Algorithm 4.1 provides a formal description. The algorithm starts a given model with weights $\theta_1$ (Line 1); then it waits for a user-submitted query (Line 3). At this point a candidate ranker is sampled from the unit sphere around the current model (Line 4), and the current and candidate model both produce a ranking for the current query (Line 5 and 6). These rankings are interleaved (Line 7) and displayed to the user (Line 8). If the interleaving method infers a preference for the candidate ranker from subsequent user interactions the current model is updated towards the candidate (Line 10), otherwise no update is performed (Line 12). Thus, the model optimized by DBGD should converge and oscillate towards an optimum.

### 4.3.2 Regret bounds of Dueling Bandit Gradient Descent

Unlike PDGD, DBGD has proven regret bounds [132], potentially providing an advantage in the form of theoretical guarantees. In this section we answer **RQ4.1** by critically looking at the assumptions which form the basis of DBGD's proven regret bounds.

The original DBGD paper [132] proved a sublinear regret under several assumptions. DBGD works with the parameterized space of ranking functions $\mathcal{W}$, that is, every $\theta \in \mathcal{W}$ is a different set of parameters for a ranking function. For this chapter we will only consider deterministic linear models because all existing OLTR work has dealt with them [40, 43, 82, 90, 111, 125, 132, 135]. But we note that the proof is easily extendable to neural networks where the output is a monotonic function applied to a linear combination of the last layer. Then there is assumed to be a concave utility function $u : \mathcal{W} \to \mathbb{R}$; since this function is concave, there should only be a single instance of weights that are optimal $\theta^*$. Furthermore, this utility function is assumed to be L-Lipschitz smooth:

$$\exists L \in \mathbb{R}, \quad \forall (\theta_a, \theta_b) \in \mathcal{W}, \quad |u(\theta_a) - u(\theta_b)| < L\|\theta_a - \theta_b\|. \tag{4.1}$$

We will show that these assumptions are *incorrect*: there is an infinite number of optimal weights, and the utility function $u$ cannot be L-Lipschitz smooth. Our proof relies on two assumptions that avoid cases where the ranking problem is trivial. First, the zero ranker is not the optimal model:

$$\theta^* \neq \mathbf{0}. \tag{4.2}$$

Second, there should be at least two models with different utility values:

$$\exists (\theta, \theta') \in \mathcal{W}, \quad u(\theta) \neq u(\theta'). \tag{4.3}$$

We will start by defining the set of rankings a model $f(\cdot, \theta)$ will produce as:

$$\mathcal{R}_D(f(\cdot, \theta)) = \{R \mid \forall (d, d') \in D, [f(d, \theta) > f(d', \theta) \to d \succ_R d']\}. \tag{4.4}$$

It is easy to see that multiplying a model with a positive scalar $\alpha > 0$ will not affect this set:

$$\forall \alpha \in \mathbb{R}_{>0}, \quad \mathcal{R}_D(f(\cdot, \theta)) = \mathcal{R}_D(\alpha f(\cdot, \theta)). \tag{4.5}$$

Consequently, the utility of both functions will be equal:

$$\forall \alpha \in \mathbb{R}_{>0}, \quad u(f(\cdot, \theta)) = u(\alpha f(\cdot, \theta)). \tag{4.6}$$

For linear models scaling weights has the same effect: $\alpha f(\cdot, \theta) = f(\cdot, \alpha\theta)$. Thus, the first assumption cannot be true since for any optimal model $f(\cdot, \theta^*)$ there is an infinite set of equally optimal models: $\{f(\cdot, \alpha\theta^*) \mid \alpha \in \mathbb{R}_{>0}\}$.

Then, regarding L-Lipschitz smoothness, using any positive scaling factor:

$$\forall \alpha \in \mathbb{R}_{>0}, \quad |u(\theta_a) - u(\theta_b)| = |u(\alpha\theta_a) - u(\alpha\theta_b)|, \tag{4.7}$$
$$\forall \alpha \in \mathbb{R}_{>0}, \quad \|\alpha\theta_a - \alpha\theta_b\| = \alpha\|\theta_a - \theta_b\|. \tag{4.8}$$

Thus the smoothness assumption can be rewritten as:

$$\exists L \in \mathbb{R}, \quad \forall \alpha \in \mathbb{R}_{>0}, \quad \forall (\theta_a, \theta_b) \in \mathcal{W}, \quad |u(\theta_a) - u(\theta_b)| < \alpha L\|\theta_a - \theta_b\|. \tag{4.9}$$

However, there is always an infinite number of values for $\alpha$ small enough to break the assumption. Therefore, we conclude that a concave L-Lipschitz smooth utility function can never exist for a deterministic linear ranking model, thus the proof for the regret bounds is not applicable when using deterministic linear models.

Consequently, the regret bounds of DBGD do not apply to the ranking problems in previous work. One may consider other models (e.g., spherical coordinate based models or stochastic ranking models), however this still means that for the simplest and most common ranking problems there are no proven regret bounds. As a result, we answer **RQ4.1** negatively, the regret bounds of DBGD do not provide a benefit over PDGD for the ranking problems in LTR.

## 4.4 Pairwise Differentiable Gradient Descent

The Pairwise Differentiable Gradient Descent (PDGD) [82] algorithm is formally described in Algorithm 4.2. PDGD interprets a ranking function $f(\cdot, \theta)$ as a probability distribution over documents by applying a Plackett-Luce model:

$$P(d|D, \theta) = \frac{e^{f(d,\theta)}}{\sum_{d' \in D} e^{f(d',\theta)}}. \tag{4.10}$$

First, the algorithm waits for a user query (Line 3), then a ranking $R$ is created by sampling documents without replacement (Line 4). Then PDGD observes clicks from the user and infers pairwise document preferences from them. All documents preceding a clicked document and the first succeeding one are assumed to be observed by the user. Preferences between clicked and unclicked observed documents are inferred by PDGD; this is a long-standing assumption in pairwise LTR [54]. We denote an *inferred* preference between documents as $d_i \succ_{\mathbf{c}} d_j$, and the probability of the model placing $d_i$ earlier than $d_j$ is denoted and calculated by:

$$P(d_i \succ d_j \mid \theta) = \frac{e^{f(d_i,\theta)}}{e^{f(d_i,\theta)} + e^{f(d_j,\theta)}}. \tag{4.11}$$

---

**Algorithm 4.2** Pairwise Differentiable Gradient Descent (PDGD).

---

1: **Input**: initial weights: $\theta_1$; scoring function: $f$; learning rate $\eta$.
2: **for** $t \leftarrow 1 \ldots \infty$ **do**
3:     $q_t \leftarrow$ receive_query($t$)               *// obtain a query from a user*
4:     $\mathbf{R}_t \leftarrow$ sample_list($f_{\theta_t}, D_{q_t}$)        *// sample list according to Eq. 4.10*
5:     $\mathbf{c}_t \leftarrow$ receive_clicks($\mathbf{R}_t$)           *// show result list to the user*
6:     $\nabla f(\cdot, \theta_t) \leftarrow \mathbf{0}$                  *// initialize gradient*
7:     **for** $d_i \succ_{\mathbf{c}} d_j \in \mathbf{c}_t$ **do**
8:         $w \leftarrow \rho(d_i, d_j, R, D)$       *// initialize pair weight (Eq. 4.13)*
9:         $w \leftarrow w \times P(d_i \succ d_j \mid \theta_t) P(d_j \succ d_i \mid \theta_t)$    *// pair gradient (Eq. 3.4)*
10:        $\nabla f(\cdot, \theta_t) \leftarrow \nabla f_{\theta_t} + w \times (f'(d_i, \theta_t) - f'(d_j, \theta_t))$   *// model gradient (Eq. 3.4)*
11:     $\theta_{t+1} \leftarrow \theta_t + \eta \nabla f(\cdot, \theta_t)$          *// update the ranking model*

---

The gradient is estimated as a sum over inferred preferences with a weight $\rho$ per pair:

$$
\begin{aligned}
\nabla f(\cdot, \theta) \\
\approx \sum_{d_i \succ_{\mathbf{c}} d_j} \rho(d_i, d_j, R, D)[\Delta P(d_i \succ d_j \mid \theta)] \quad (4.12) \\
= \sum_{d_i \succ_{\mathbf{c}} d_j} \rho(d_i, d_j, R, D) P(d_i \succ d_j \mid \theta) P(d_j \succ d_i \mid \theta)(f'(d_i, \theta) - f'(d_j, \theta)).
\end{aligned}
$$

After computing the gradient (Line 10), the model is updated accordingly (Line 11). This will change the distribution (Equation 4.10) towards the inferred preferences. This distribution models the confidence over which documents should be placed first; the exploration of PDGD is naturally guided by this confidence and can vary per query.

The weighting function $\rho$ is used to make the gradient of PDGD unbiased w.r.t. document pair preferences. It uses the reverse pair ranking: $R^*(d_i, d_j, R)$, which is the same ranking as $R$ but with the document positions of $d_i$ and $d_j$ swapped. Then $\rho$ is the ratio between the probability of $R$ and $R^*$:

$$
\rho(d_i, d_j, R, D) = \frac{P(R^*(d_i, d_j, R) \mid D)}{P(R \mid D) + P(R^*(d_i, d_j, R) \mid D)}. \quad (4.13)
$$

In Chapter 3, the weighted gradient is proven to be unbiased w.r.t. document pair preferences under certain assumptions about the user. Here, this unbiasedness is defined by being able to rewrite the gradient as:

$$
E[\Delta f(\cdot, \theta)] = \sum_{(d_i, d_j) \in D} \alpha_{ij}(f'(\mathbf{d}_i, \theta) - f'(\mathbf{d}_j, \theta)), \quad (4.14)
$$

and the sign of $\alpha_{ij}$ agreeing with the preference of the user:

$$
sign(\alpha_{ij}) = sign(relevance(d_i) - relevance(d_j)). \quad (4.15)
$$

The proof in Chapter 3 only relies on the difference in the probabilities of inferring a preference: $d_i \succ_{\mathbf{c}} d_j$ in $R$ and the opposite preference $d_j \succ_{\mathbf{c}} d_i$ in $R^*(d_i, d_j, R)$. The

Table 4.1: Click probabilities for simulated *perfect* or *almost random* behavior.

| | $P(click(d) \mid relevance(d), observed(d))$ | | | | |
|---|---|---|---|---|---|
| *relevance(d)* | 0 | 1 | 2 | 3 | 4 |
| *perfect* | 0.00 | 0.20 | 0.40 | 0.80 | 1.00 |
| *almost random* | 0.40 | 0.45 | 0.50 | 0.55 | 0.60 |

proof relies on the sign of this difference to match the user's preference:

$$sign(P(d_i \succ_{\mathbf{c}} d_j \mid R) - P(d_j \succ_{\mathbf{c}} d_i \mid R^*))$$
$$= sign(relevance(d_i) - relevance(d_j)). \quad (4.16)$$

As long as Equation 4.16 is true, Equation 4.14 and 4.15 hold as well. Interestingly, this means that other assumptions about the user can be made than in Chapter 3, and other variations of PDGD are possible, e.g., the algorithm could assume that all documents are observed and the proof still holds.

Chapter 3 reports large improvements over DBGD, however these improvements were observed under simulated cascading user models. This means that the assumption that PDGD makes about which documents are observed are always true. As a result, it is currently unclear whether the method is really better in cases where the assumption does not hold.

## 4.5 Experiments

In this section we detail the experiments that were performed to answer the research questions in Section 4.1.[1]

### 4.5.1 Datasets

Our experiments are performed over three large labelled datasets from commercial search engines, the largest publicly available LTR datasets. These datasets are the *MLSR-WEB10K* [95], *Yahoo! Webscope* [17], and *Istella* [27] datasets. Each contains a set of queries with corresponding preselected document sets. Query-document pairs are represented by feature vectors and five-grade relevance annotations ranging from *not relevant* (0) to *perfectly relevant* (4). Together, the datasets contain over 29,900 queries and between 136 and 700 features per representation.

### 4.5.2 Simulating user behavior

In order to simulate user behavior we partly follow the standard setup for OLTR [38, 40, 90, 111, 137]. At each step a user issued query is simulated by uniformly sampling from the datasets. The algorithm then decides what result list to display to the user, the

---

[1]The resources for reproducing the experiments in this chapter are available at `https://github.com/HarrieO/OnlineLearningToRank`

result list is limited to $k = 10$ documents. Then user interactions are simulated using click models [20]. Past OLTR work has only considered *cascading click models* [36]; in contrast, we also use *non-cascading click models*. The probability of a click is conditioned on relevance and observance:

$$P(click(d) \mid relevance(d), observed(d)). \tag{4.17}$$

We use two levels of noise to simulate *perfect* user behavior and *almost random* behavior [39], Table 4.1 lists the probabilities of both. The *perfect* user observes all documents, never clicks on anything non-relevant, and always clicks on the most relevant documents. Two variants of *almost random* behavior are used. The first is based on cascading behavior, here the user first observes the top document, then decides to click according to Table 4.1. If a click occurs, then, with probability $P(stop \mid click) = 0.5$ the user stops looking at more documents, otherwise the process continues on the next document. The second *almost random* behavior is simulated in a non-cascading way; here we follow [58] and model the observing probabilities as:

$$P(observed(d) \mid rank(d)) = \frac{1}{rank(d)}. \tag{4.18}$$

The important distinction is that it is safe to assume that the cascading user has observed all documents ranked before a click, while this is not necessarily true for the non-cascading user. Since PDGD makes this assumption, testing under both models can show us how much of its performance relies on this assumption. Furthermore, the *almost random* model has an extreme level of noise and position bias compared to the click models used in previous OLTR work [40, 90, 111], and we argue it simulates an (almost) worst-case scenario.

## 4.5.3   Experimental runs

In our experiments we simulate runs consisting of 1,000,000 impressions; each run was repeated 125 times under each of the three click models. PDGD was run with $\eta = 0.1$ and zero initialization, DBGD was run using Probabilistic Interleaving [90] with zero initialization, $\eta = 0.001$, and the unit sphere with $\delta = 1$. Other variants like Multileave Gradient Descent [111] are not included; previous work has shown that their performance matches that of regular DBGD after around 30,000 impressions [82, 90, 111]. The initial boost in performance comes at a large computational cost, though, as the fastest approaches keep track of at least 50 ranking models [90], which makes running long experiments extremely impractical. Instead, we introduce a novel oracle version of DBGD, where, instead of interleaving, the NDCG values on the current query are calculated and the highest scoring model is selected. This simulates a hypothetical perfect interleaving method, and we argue that the performance of this oracle run indicates what the upper bound on DBGD performance is.

Performance is measured by NDCG@10 on a held-out test set, a two-sided t-test is performed for significance testing. We do not consider the user experience during training, because Chapter 3 has already investigated this aspect thoroughly.

## 4.6 Experimental Results and Analysis

Recall that in Section 4.3.2 we have already provided a negative answer to **RQ4.1**: the regret bounds of DBGD do not provide a benefit over PDGD for the common ranking problem in LTR. In this section we present our experimental results and answer **RQ4.2** (whether the advantages of PDGD over DBGD of previous work generalize to extreme levels of noise and bias) and **RQ4.3** (whether the performance of PDGD is reproducible under non-cascading user behavior).

Our main results are presented in Table 4.2. Additionally, Figure 4.1 displays the average performance over 1,000,000 impressions. First, we consider the performance of DBGD; there is a substantial difference between its performance under the *perfect* and *almost random* user models on all datasets. Thus, it seems that DBGD is strongly affected by noise and bias in interactions; interestingly, there is little difference between performance under the cascading and non-cascading behavior. On all datasets the *oracle* version of DBGD performs significantly better than DBGD under *perfect* user behavior. This means there is still room for improvement and hypothetical improvements in, e.g., interleaving could lead to significant increases in long-term DBGD performance.

Next, we look at the performance of PDGD; here, there is also a significant difference between performance under the *perfect* and *almost random* user models on all datasets. However, the effect of noise and bias is very limited compared to DBGD, and this difference at 1,000,000 impressions is always less than 0.03 NDCG on any dataset.

To answer **RQ4.2**, we compare the performance of DBGD and PDGD. Across all datasets, when comparing DBGD and PDGD under the same levels of interaction noise and bias, the performance of PDGD is significantly better in every case. Furthermore, PDGD under the *perfect* user model significantly outperforms the *oracle* run of DBGD, despite the latter being able to directly observe the NDCG of rankers on the current query. Moreover, when comparing PDGD's performance under the *almost random* user model with DBGD under the *perfect* user model, we see the differences are limited and in both directions. Thus, even under ideal circumstances DBGD does not consistently outperform PDGD under extremely difficult circumstances. As a result, we answer **RQ4.2** positively: our results strongly indicate that the performance of PDGD is considerably better than DBGD and that these findings generalize from ideal circumstances to settings with extreme levels of noise and bias.

Finally, to answer **RQ4.3**, we look at the performance under the two *almost random* user models. Surprisingly, there is no clear difference between the performance of PDGD under *cascading* and *non-cascading* user behavior. The differences are small and per dataset it differs which circumstances are slightly preferred. Therefore, we answer **RQ4.3** positively: the performance of PDGD is reproducible under *non-cascading* user behavior.

## 4.7 Conclusion

In this chapter, we have reproduced and generalized findings about the relative performance of Dueling Bandit Gradient Descent (DBGD) and Pairwise Differentiable Gradient Descent (PDGD). Our results show that the performance of PDGD is repro-

Table 4.2: Performance (NDCG@10) after 1,000,000 impressions for DBGD and PDGD under a *perfect* click model and two almost-random click models: *cascading* and *non-cascading*, and DBGD with an *oracle* comparator. Significant improvements and losses (p < 0.01) between DBGD and PDGD are indicated by ▲, ▼, and ○ (no significant difference). Indications are in order of: *oracle*, *perfect*, *cascading*, and *non-cascading*.

| | Yahoo | MSLR | Istella |
|---|---|---|---|
| | *Dueling Bandit Gradient Descent* | | |
| *oracle* | 0.744 (0.001) ▼ ▲ ▲ | 0.438 (0.004) ▼ ▲ ▲ | 0.584 (0.001) ▼ ▲ ▲ |
| *perfect* | 0.730 (0.002) ▼ ○ ○ | 0.426 (0.004) ▼ ▲ ▲ | 0.554 (0.002) ▼ ▼ ▼ |
| *cascading* | 0.696 (0.008) ▼ ▼ ▼ | 0.320 (0.006) ▼ ▼ ▼ | 0.415 (0.014) ▼ ▼ ▼ |
| *non-cascading* | 0.692 (0.010) ▼ ▼ ▼ | 0.320 (0.014) ▼ ▼ ▼ | 0.422 (0.014) ▼ ▼ ▼ |
| | *Pairwise Differentiable Gradient Descent* | | |
| *perfect* | 0.752 (0.001) ▲ ▲ ▲ ▲ | 0.442 (0.003) ▲ ▲ ▲ ▲ | 0.592 (0.000) ▲ ▲ ▲ ▲ |
| *cascading* | 0.730 (0.003) ▼ ○ ▲ ▲ | 0.420 (0.007) ▼ ▼ ▲ ▲ | 0.563 (0.003) ▼ ▲ ▲ ▲ |
| *non-cascading* | 0.729 (0.003) ▼ ○ ▲ ▲ | 0.424 (0.005) ▼ ▼ ▲ ▲ | 0.570 (0.003) ▼ ▲ ▲ ▲ |

ducible under non-cascading user behavior. Furthermore, PDGD outperforms DBGD in both *ideal* and extremely *difficult* circumstances with high levels of noise and bias. Moreover, the performance of PDGD in extremely *difficult* circumstances is comparable to that of DBGD in *ideal* circumstances. Additionally, we have shown that the regret bounds of DBGD are not applicable to the common ranking problem in LTR. In summary, our results strongly confirm the previous finding that PDGD consistently outperforms DBGD, and generalizes this conclusion to circumstances with extreme levels of noise and bias.

With these findings we can answer **RQ3** mostly negatively: the theory behind DBGD is not sound for the common deterministic ranking problem, moreover, DBGD has extremely poor performance when compared to the PDGD method under varying conditions. Consequently, there appears to be no advantage to using DBGD over PDGD in either theoretical or empirical terms. In addition, a decade of OLTR work has attempted to extend DBGD in numerous ways without leading to any measurable long-term improvements. Together, this suggests that the general approach of DBGD based methods, i.e., sampling models and comparing with online evaluation, is not an effective way of optimizing ranking models. Although the PDGD method considerably outperforms the DBGD approach, we currently do not have a theoretical explanation for this difference. Thus it seems plausible that a more effective OLTR method could be derived, if the theory behind the effectiveness of OLTR methods is better understood. Due to this potential and the current lack of regret bounds applicable to OLTR, we argue that a theoretical analysis of OLTR would make a very valuable future contribution to the field.

Finally, we consider the limitations of the comparison in this chapter. As is standard in OLTR our results are based on simulated user behavior. These simulations provide valuable insights: they enable direct control over biases and noise, and evaluation can

be performed at each time step. In this chapter, the generalizability of this setup was pushed the furthest by varying the conditions to the extremely difficult. It appears unlikely that more reliable conclusions can be reached from simulated behavior. Thus we argue that the most valuable future comparisons would be in experimental settings with real users. Furthermore, with the performance improvements of PDGD the time seems right for evaluating the effectiveness of OLTR in real-world applications.

The limited theoretical guarantees regarding OLTR methods, prompted the second part of this thesis where we consider counterfactual LTR. In contrast with OLTR, counterfactual LTR methods are founded on assumed models of user behavior and are proven to unbiasedly optimize ranking metrics if the assumed models are correct. Despite these theoretical strengths, empirical comparisons in previous work show that PDGD is more robust than existing counterfactual LTR methods. In Chapter 8 we introduce a counterfactual LTR method that can reach the same levels of performance as PDGD when applied online.

Figure 4.1: Performance (NDCG@10) on held-out data from Yahoo (top), MSLR (center), Istella (bottom) datasets, under the *perfect*, and *almost random* user models: cascading (casc.) and non-cascading (non-casc.). The shaded areas display the standard deviation.
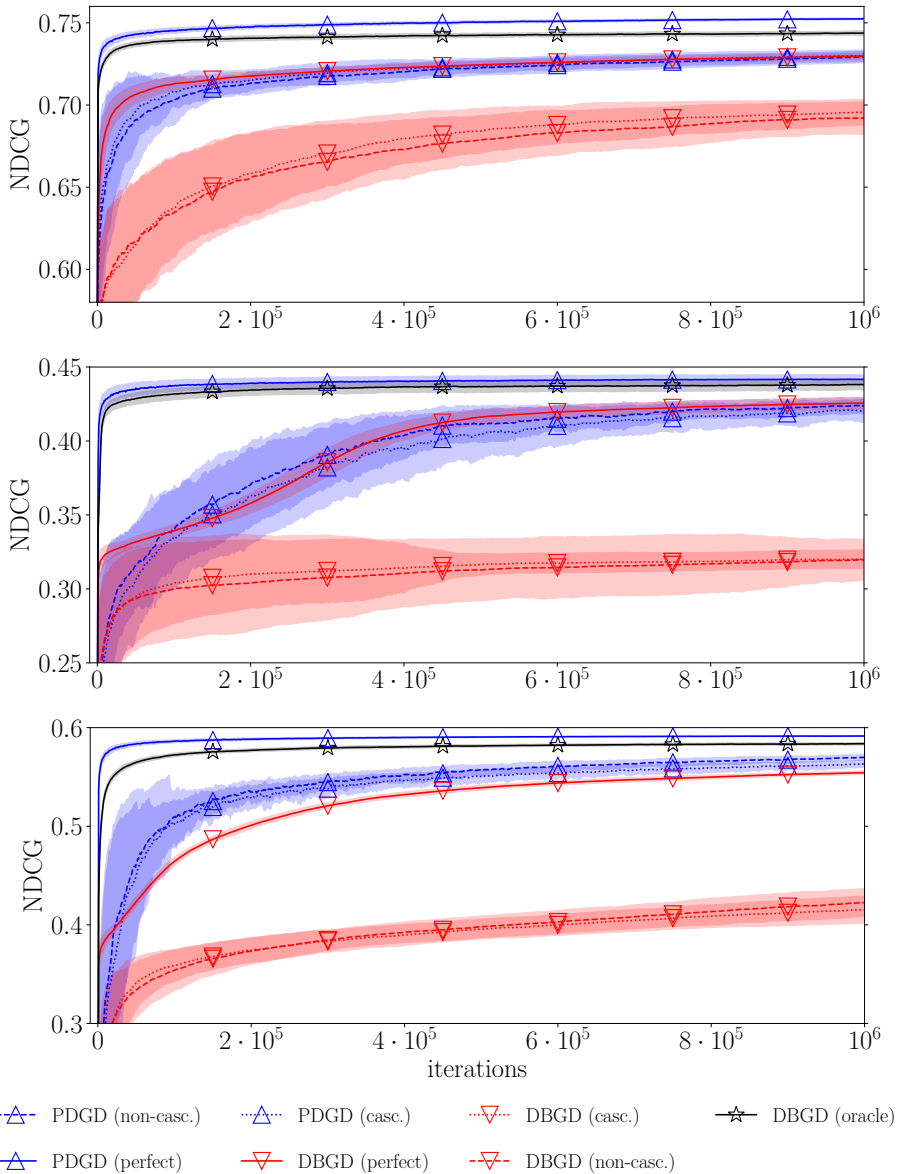
# 4.A   Notation Reference for Chapter 4

| Notation | Description |
| --- | --- |
| $t$ | a timestep |
| $q$ | a user-issued query |
| $d$, $d_k$, $d_l$ | document |
| $\mathbf{d}$ | feature representation of a query-document pair |
| $D$ | set of documents |
| $R$ | ranked list |
| $I_t$ | an interleaved result list |
| $R^*$ | the reversed pair ranking $R^*(d_k, d_l, R)$ |
| $\rho$ | preference pair weighting function |
| $\theta$ | parameters of the ranking model |
| $f_\theta(\cdot)$ | ranking model with parameters $\theta$ |
| $f(\mathbf{d}_k)$ | ranking score for a document from model |
| $\mathbf{c}_t$ | a binary vector representing the clicks at timestep $t$ |

**Part II**

# A Single Framework for Online and Counterfactual Learning to Rank

# 5

# Policy-Aware Counterfactual Learning to Rank for Top-$k$ Rankings

Counterfactual Learning to Rank (LTR) methods optimize ranking systems using logged user interactions that contain interaction biases. Existing methods are only unbiased if users are presented with all relevant items in every ranking. However, in prevalent top-$k$ ranking settings not all items can be displayed at once. Therefore, there is currently no existing counterfactual unbiased LTR method for top-$k$ rankings. In this chapter we address this limitation by asking the thesis research question:

**RQ4**  Can counterfactual LTR be extended to top-$k$ ranking settings?

We introduce a novel policy-aware counterfactual estimator for LTR metrics that can account for the effect of a stochastic logging policy. We prove that the policy-aware estimator is unbiased if every relevant item has a non-zero probability to appear in the top-$k$ ranking. Our experimental results show that the performance of our estimator is not affected by the size of $k$: for any $k$, the policy-aware estimator reaches the same retrieval performance while learning from top-$k$ feedback as when learning from feedback on the full ranking.

While the policy-aware estimator allows us to learn from top-$k$ feedback, there is no theoretically-grounded way to optimize for top-$k$ ranking metrics. Furthermore, existing counterfactual LTR work has mostly used novel loss functions for optimization, which are quite different from those used in supervised LTR. This lead us to ask the following thesis research question:

**RQ5**  Is it possible to apply state-of-the-art supervised LTR to the counterfactual LTR problem?

In this chapter, we also introduce novel extensions of supervised LTR methods to perform counterfactual LTR and to optimize top-$k$ metrics. Together, our contributions introduce the first policy-aware unbiased LTR approach that learns from top-$k$ feedback and optimizes top-$k$ metrics. As a result, counterfactual LTR is now applicable to the very prevalent top-$k$ ranking setting in search and recommendation.

---

This chapter was published as [86]. Appendix 5.A gives a reference for the notation used in this chapter.

# 5.1   Introduction

LTR optimizes ranking systems to provide high quality rankings. Interest in LTR from user interactions has greatly increased in recent years with the introduction of unbiased LTR methods [58, 127]. The potential for learning from logged user interactions is great: user interactions provide valuable implicit feedback while also being cheap and relatively easy to acquire at scale [57]. However, interaction logs also contain large amounts of bias, which is the result of both user behavior and the ranker used during logging. For instance, users are more likely to examine items at the top of rankings, consequently the display position of an item heavily affects the number of interactions it receives [128]. This effect is called *position bias* and it is very dominant when learning from interactions with rankings. Naively ignoring it during learning can be detrimental to ranking performance, as the learning process is strongly impacted by what rankings were displayed during logging instead of *true* user preferences. The goal of unbiased LTR methods is to optimize a ranker w.r.t. the *true* user preferences, consequently, they have to account and correct for such forms of bias.

Previous work on unbiased LTR has mainly focussed on accounting for position bias through counterfactual learning [5, 58, 127]. The prevalent approach models the probability of a user examining an item in a displayed ranking. This probability can be inferred from user interactions [4, 5, 58, 127, 128] and corrected for using *inverse propensity scoring*. As a result, these methods optimize a loss that in expectation is unaffected by the examination probabilities during logging, hence it is unbiased w.r.t. position bias.

This approach has been applied effectively in various ranking settings, including search for scientific articles [58], email [127] or other personal documents [128]. However, a limitation of existing approaches is that in every logged ranking they require every relevant item to have a non-zero chance of being examined [16, 58]. In this chapter, we focus on top-$k$ rankings where the number of displayed items is systematically limited. These rankings can display at most $k$ items, making it practically unavoidable that relevant items are missing. Consequently, existing counterfactual LTR methods are not unbiased in these settings. We recognize this problem as *item-selection bias* introduced by the selection of (only) $k$ items to display. This is especially concerning since top-$k$ rankings are quite prevalent, e.g., in recommendation [26, 48], mobile search [9, 124], query autocompletion [14, 127, 128], and digital assistants [112].

Our main contribution is a novel policy-aware estimator for counterfactual LTR that accounts for both a stochastic logging policy and the users' examination behavior. Our policy-aware approach can be viewed as a generalization of the existing counterfactual LTR framework [2, 58]. We prove that our policy-aware approach performs unbiased LTR and evaluation while learning from top-$k$ feedback. Our experimental results show that while our policy-aware estimator is unaffected by the choice of $k$, the existing policy-oblivious approach is strongly affected even under large values of $k$. For instance, optimization with the policy-aware estimator on top-5 feedback reaches the same performance as when receiving feedback on all results. Furthermore, because top-$k$ metrics are the only relevant metrics in top-$k$ rankings, we also propose extensions to traditional LTR approaches that are proven to optimize top-$k$ metrics unbiasedly and introduce a pragmatic way to choose optimally between available loss functions.

This chapter is based around two main contributions:

1. A novel estimator for unbiased LTR from top-$k$ feedback.

2. Unbiased losses that optimize bounds on top-$k$ LTR metrics.

To the best of our knowledge, our policy-aware estimator is the first estimator that is unbiased in top-$k$ ranking settings.

## 5.2   Background

In this section we discuss supervised LTR and counterfactual LTR [58].

### 5.2.1   Supervised learning to rank

The goal of LTR is to optimize ranking systems w.r.t. specific ranking metrics. Ranking metrics generally involve items $d$, their relevance $r$ w.r.t. a query $q$, and their position in the ranking $R$ produced by the system. We will optimize the *Empirical Risk* [121] over the set of queries $Q$, with a loss $\Delta(R_i \mid q_i, r)$ for a single query $q_i$:

$$\mathcal{L} = \frac{1}{|Q|} \sum_{q_i \in Q} \Delta(R_i \mid q_i, r). \tag{5.1}$$

For simplicity we assume that relevance is binary: $r(q, d) \in \{0, 1\}$; for brevity we write: $r(q, d) = r(d)$. Then, ranking metrics commonly take the form of a sum over items:

$$\Delta(R \mid q, r) = \sum_{d \in R} \lambda(d \mid R) \cdot r(d), \tag{5.2}$$

where $\lambda$ can be chosen for a specific metric, e.g., for Average Relevance Position (ARP) or Discounted Cumulative Gain (DCG):

$$\lambda^{ARP}(d \mid R) = \text{rank}(d \mid R), \tag{5.3}$$

$$\lambda^{DCG}(d \mid R) = -\log_2\left(1 + \text{rank}(d \mid R)\right)^{-1}. \tag{5.4}$$

In a so-called *full-information* setting, where the relevance values $r$ are known, optimization can be done through traditional LTR methods [13, 54, 75, 129].

### 5.2.2   Counterfactual learning to rank

Optimizing a ranking loss from the implicit feedback in interaction logs requires a different approach from supervised LTR. We will assume that clicks are gathered using a logging policy $\pi$ with the probability of displaying ranking $\bar{R}$ for query $q$ denoted as $\pi(\bar{R} \mid q)$. Let $o_i(d) \in \{0, 1\}$ indicate whether $d$ was examined by a user at interaction $i$ and $o_i(d) \sim P(o(d) \mid q_i, r, \bar{R}_i)$. Furthermore, we assume that users click on all relevant items they observe and nothing else: $c_i(d) = \mathbb{1}[r(d) \wedge o_i(d)]$. Our goal is to find an

estimator $\hat{\Delta}$ that provides an unbiased estimate of the actual loss; for $N$ interactions this estimate is:

$$\hat{\mathcal{L}} = \frac{1}{N} \sum_{i=1}^{N} \hat{\Delta}(R_i \mid q_i, \bar{R}_i, \pi, c_i). \tag{5.5}$$

We write $R_i$ for the ranking produced by the system for which the loss is being computed, while $\bar{R}_i$ is the ranking that was displayed when logging interaction $i$. For brevity we will drop $i$ from our notation when only a single interaction is involved. A naive estimator could simply consider every click to indicate relevance:

$$\hat{\Delta}_{naive}(R \mid q, c) = \sum_{d:c(d)=1} \lambda(d \mid R). \tag{5.6}$$

Taking the expectation over the displayed ranking and observance variables results in the following expected loss:

$$\mathbb{E}_{o,\bar{R}}\left[\hat{\Delta}_{naive}(R \mid q, c)\right]$$

$$= \mathbb{E}_{o,\bar{R}}\left[\sum_{d:c(d)=1} \lambda(d \mid R)\right] = \mathbb{E}_{o,\bar{R}}\left[\sum_{d \in R} \lambda(d \mid R) \cdot c(d)\right]$$

$$= \mathbb{E}_{o,\bar{R}}\left[\sum_{d \in R} o(d) \cdot \lambda(d \mid R) \cdot r(d)\right] \tag{5.7}$$

$$= \mathbb{E}_{\bar{R}}\left[\sum_{d \in R} P\left(o(d) = 1 \mid q, r, \bar{R}\right) \cdot \lambda(d \mid R) \cdot r(d)\right]$$

$$= \sum_{\bar{R} \in \pi(\cdot \mid q)} \pi(\bar{R} \mid q) \cdot \sum_{d \in R} P\left(o(d) = 1 \mid q, r, \bar{R}\right) \cdot \lambda(d \mid R) \cdot r(d).$$

Here, the effect of position bias is very clear; in expectation, items are weighted according to their probability of being examined. Furthermore, it shows that examination probabilities are determined by both the logging policy $\pi$ and user behavior $P(o(d) \mid q, r, \bar{R})$.

In order to avoid the effect of position bias, Joachims et al. [58] introduced an inverse-propensity-scoring estimator in the same vain as previous work by Wang et al. [127]. The main idea behind this estimator is that if the examination probabilities are known, then they can be corrected for per click:

$$\hat{\Delta}_{oblivious}(R \mid q, c, \bar{R}) = \sum_{d:c(d)=1} \frac{\lambda(d \mid R)}{P\left(o(d) = 1 \mid q, r, \bar{R}\right)}. \tag{5.8}$$

In contrast to the naive estimator (Eq. 5.6), this policy-oblivious estimator (Eq. 5.8) can

provide an unbiased estimate of the loss:

$$
\mathbb{E}_{o,\bar{R}}\left[\hat{\Delta}_{oblivious}\left(R \mid q, c, \bar{R}\right)\right]
$$

$$
= \mathbb{E}_{o,\bar{R}}\left[\sum_{d:c(d)=1}\frac{\lambda\left(d \mid R\right)}{P\left(o(d)=1 \mid q, r, \bar{R}\right)}\right]
$$

$$
= \sum_{d \in R}\mathbb{E}_{o,\bar{R}}\left[\frac{o(d) \cdot \lambda\left(d \mid R\right) \cdot r(d)}{P\left(o(d)=1 \mid q, r, \bar{R}\right)}\right] \tag{5.9}
$$

$$
= \sum_{d \in R}\mathbb{E}_{\bar{R}}\left[\frac{P\left(o(d)=1 \mid q, r, \bar{R}\right) \cdot \lambda\left(d \mid R\right) \cdot r(d)}{P\left(o(d)=1 \mid q, r, \bar{R}\right)}\right]
$$

$$
= \sum_{d \in R}\lambda\left(d \mid R\right) \cdot r(d) = \Delta(R \mid q, r).
$$

We note that the last step assumes $P\left(o(d)=1 \mid q, r, \bar{R}\right) > 0$, and that only relevant items $r(d) = 1$ contribute to the estimate [58]. Therefore, this estimator is unbiased as long as the examination probabilities are positive for every relevant item:

$$
\forall d, \forall \bar{R} \in \pi(\cdot \mid q)\left[r(d)=1 \rightarrow P\left(o(d)=1 \mid q, r, \bar{R}\right) > 0\right]. \tag{5.10}
$$

Intuitively, this condition exists because propensity weighting is applied to items clicked in the displayed ranking and items that cannot be observed can never receive clicks. Thus, there are no clicks that can be weighted more heavily to adjust for the zero observance probability of an item.

An advantageous property of the policy-oblivious estimator $\hat{\Delta}_{oblivious}$ is that the logging policy $\pi$ does not have to be known. That is, as long as Condition 5.10 is met, it works regardless of how interactions were logged. Additionally, Joachims et al. [58] proved that it is still unbiased under click noise. Virtually all recent counterfactual LTR methods use the policy-oblivious estimator for LTR optimization [3–5, 58, 127, 128].

## 5.3 Learning from Top-$k$ Feedback

In this section we explain why the existing policy-oblivious counterfactual LTR framework is not applicable to top-$k$ rankings. Subsequently, we propose a novel solution through policy-aware propensity scoring that takes the logging policy into account.

### 5.3.1 The problem with top-$k$ feedback

An advantage of the existing policy-oblivious estimator for counterfactual LTR described in Section 5.2.2 is that the logging policy does not need to be known, making its application easier. However, the policy-oblivious estimator is only unbiased when Condition 5.10 is met: every relevant item has a non-zero probability of being observed in every ranking displayed during logging.

We recognize that in top-$k$ rankings, where only $k$ items can be displayed, relevant items may systematically *lack* non-zero examination probabilities. This happens because items outside the top-$k$ cannot be examined by the user:

$$\forall d, \forall \bar{R} \left[ \mathrm{rank}\big(d \mid \bar{R}\big) > k \rightarrow P\big(o(d) = 1 \mid q, r, \bar{R}\big) = 0 \right]. \qquad (5.11)$$

In most top-$k$ ranking settings it is very unlikely that Condition 5.10 is satisfied; If $k$ is very small, the number of relevant items is large, or if the logging policy $\pi$ is ineffective at retrieving relevant items, it is unlikely that all relevant items will be displayed in the top-$k$ positions. Moreover, for a small value of $k$ the performance of the logging policy $\pi$ has to be near ideal for all relevant items to be displayed. We call this effect *item-selection bias*, because in this setting the logging ranker makes a selection of which $k$ items to display, in addition to the order in which to display them (position bias). The existing policy-oblivious estimator for counterfactual LTR (as described in Section 5.2.2) cannot correct for item-selection bias when it occurs, and can thus be affected by this bias when applied to top-$k$ rankings.

## 5.3.2 Policy-aware propensity scoring

Item-selection bias is inevitable in a single top-$k$ ranking, due to the limited number of items that can be displayed. However, across multiple top-$k$ rankings more than $k$ items could be displayed if the displayed rankings differ enough. Thus, a stochastic logging-policy could provide every item with a non-zero probability to appear in the top-$k$ ranking. Then, the probability of examination can be calculated as an expectation over the displayed ranking:

$$
\begin{aligned}
P\left(o(d) = 1 \mid q, r, \pi\right) &= \mathbb{E}_{\bar{R}} \left[ P\big(o(d) = 1 \mid q, r, \bar{R}\big) \right] \qquad (5.12) \\
&= \sum_{\bar{R} \in \pi(\cdot \mid q)} \pi\big(\bar{R} \mid q\big) \cdot P\big(o(d) = 1 \mid q, r, \bar{R}\big).
\end{aligned}
$$

This policy-dependent examination probability can be non-zero for all items, even if all items cannot be displayed in a single top-$k$ ranking. Naturally, this leads to a *policy-aware* estimator:

$$\hat{\Delta}_{aware}\left(R \mid q, c, \pi\right) = \sum_{d : c(d) = 1} \frac{\lambda(d \mid R)}{P\big(o(d) = 1 \mid q, r, \pi\big)}. \qquad (5.13)$$

By basing the propensity on the policy instead of the individual rankings, the policy-aware estimator can correct for zero observance probabilities in some displayed rankings by more heavily weighting clicks on other displayed rankings with non-zero observance probabilities. Thus, if a click occurs on an item that the logging policy rarely displays in a top-$k$ ranking, this click may be weighted more heavily than a click on an item that is displayed in the top-$k$ very often. In contrast, the policy-oblivious approach only corrects for the observation probability for the displayed ranking in which the click occurred, thus it does not correct for the fact that an item may be missing from the top-$k$ in other displayed rankings.

In expectation, the policy-aware estimator provides an unbiased estimate of the ranking loss:

$$\mathbb{E}_{o,\bar{R}}\left[\hat{\Delta}_{aware}\left(R \mid q, c, \pi\right)\right]$$

$$= \mathbb{E}_{o,\bar{R}}\left[\sum_{d:c(d)=1} \frac{\lambda\left(d \mid R\right)}{P\left(o(d) = 1 \mid q, r, \pi\right)}\right]$$

$$= \sum_{d \in R} \mathbb{E}_{o,\bar{R}}\left[\frac{o(d) \cdot \lambda\left(d \mid R\right) \cdot r(d)}{\sum_{\bar{R}' \in \pi(\cdot|q)} \pi\left(\bar{R}' \mid q\right) \cdot P\left(o(d) = 1 \mid q, r, \bar{R}'\right)}\right]$$

$$= \sum_{d \in R} \mathbb{E}_{\bar{R}}\left[\frac{P\left(o(d) = 1 \mid q, r, \bar{R}\right) \cdot \lambda\left(d \mid R\right) \cdot r(d)}{\sum_{\bar{R}' \in \pi(\cdot|q)} \pi\left(\bar{R}' \mid q\right) \cdot P\left(o(d) = 1 \mid q, r, \bar{R}'\right)}\right] \quad (5.14)$$

$$= \sum_{d \in R} \frac{\sum_{\bar{R} \in \pi(\cdot|q)} \pi\left(\bar{R} \mid q\right) \cdot P\left(o(d) = 1 \mid q, r, \bar{R}\right) \cdot \lambda\left(d \mid R\right) \cdot r(d)}{\sum_{\bar{R}' \in \pi(\cdot|q)} \pi\left(\bar{R}' \mid q\right) \cdot P\left(o(d) = 1 \mid q, r, \bar{R}'\right)}$$

$$= \sum_{d \in R} \lambda\left(d \mid R\right) \cdot r(d)$$

$$= \Delta\left(R \mid q, r\right).$$

In contrast to the policy-oblivious approach (Section 5.2.2), this proof is sound as long as every relevant item has a non-zero probability of being examined under the logging policy $\pi$:

$$\forall d \left[r(d) = 1 \rightarrow \sum_{\bar{R} \in \pi(\cdot|q)} \pi\left(\bar{R} \mid q\right) \cdot P\left(o(d) = 1 \mid q, r, \bar{R}\right) > 0\right]. \quad (5.15)$$

It is easy to see that Condition 5.10 implies Condition 5.15, in other words, for all settings where the policy-oblivious estimator (Eq. 5.8) is unbiased, the policy-aware estimator (Eq. 5.13) is also unbiased. Conversely, Condition 5.15 does not imply Condition 5.10, thus there are cases where the policy-aware estimator is unbiased but the policy-oblivious estimator is not guaranteed to be.

To better understand for which policies Condition 5.15 is satisfied, we introduce a substitute Condition 5.16:

$$\forall d \left[r(d) = 1 \rightarrow \exists \bar{R} \left[\pi\left(\bar{R} \mid q\right) > 0 \wedge P\left(o(d) = 1 \mid q, r, \bar{R}\right) > 0\right]\right]. \quad (5.16)$$

Since Condition 5.16 is equivalent to Condition 5.15, we see that the policy-aware estimator is unbiased for any logging-policy that provides a non-zero probability for every relevant item to appear in a position with a non-zero examination probability. Thus to satisfy Condition 5.16 in a top-$k$ ranking setting, every relevant item requires a non-zero probability of being displayed in the top-$k$.

As long as Condition 5.16 is met, a wide variety of policies can be chosen according to different criteria. Moreover, the policy can be deterministic if $k$ is large enough to display every relevant item. Similarly, the policy-oblivious estimator can be seen as a

special case of the policy-aware estimator where the policy is deterministic (or assumed to be). The big advantage of our policy-aware estimator is that it is applicable to a much larger number of settings than the existing policy-oblivious estimator, including those were feedback is only received on the top-$k$ ranked items.

### 5.3.3 Illustrative example

To better understand the difference between the policy-oblivious and policy-aware estimators, we introduce an illustrative example that contrasts the two. We consider a single query $q$ and a logging policy $\pi$ that chooses between two rankings to display: $\bar{R}_1$ and $\bar{R}_2$, with: $\pi(\bar{R}_1 \mid q) > 0$; $\pi(\bar{R}_2 \mid q) > 0$; and $\pi(\bar{R}_1 \mid q) + \pi(\bar{R}_2 \mid q) = 1$. Then for a generic estimator we consider how it treats a single relevant item $d_n$ with $r(d_n) \neq 0$ using the expectation:

$$
\mathbb{E}_{o,\bar{R}} \left[ \frac{c(d_n) \cdot \lambda(d_n \mid R)}{\rho(o(d_n) = 1 \mid q, d_n, \bar{R}, \pi)} \right] = \lambda(d_n \mid R) \cdot r(d_n) \cdot
$$
$$
\left( \frac{\pi(\bar{R}_1|q) \cdot P(o(d_n) = 1|q, r, \bar{R}_1)}{\rho(o(d_n) = 1 \mid q, d_n, \bar{R}_1, \pi)} + \frac{\pi(\bar{R}_2|q) \cdot P(o(d_n) = 1|q, r, \bar{R}_2)}{\rho(o(d_n) = 1 \mid q, d_n, \bar{R}_2, \pi)} \right), \tag{5.17}
$$

where the propensity function $\rho$ can be chosen to match either the policy-oblivious (Eq. 5.8) or policy-aware (Eq. 5.13) estimator.

First, we examine the situation where $d_n$ appears in the top-$k$ of both rankings $\bar{R}_1$ and $\bar{R}_2$, thus it has a positive observance probability in both cases: $P(o(d_n) = 1 \mid q, r, \bar{R}_1) > 0$ and $P(o(d_n) = 1 \mid q, r, \bar{R}_2) > 0$. Here, the policy-oblivious estimator $\hat{\Delta}_{oblivious}$ (Eq. 5.8) removes the effect of observation bias by adjusting for the observance probability per displayed ranking:

$$
\left( \frac{\pi(\bar{R}_1|q) \cdot P(o(d_n) = 1|q, r, \bar{R}_1)}{P(o(d_n) = 1 \mid q, r, \bar{R}_1)} + \frac{\pi(\bar{R}_2|q) \cdot P(o(d_n) = 1|q, r, \bar{R}_2)}{P(o(d_n) = 1 \mid q, r, \bar{R}_2)} \right) \tag{5.18}
$$
$$
\cdot \lambda(d_n \mid R) \cdot r(d_n) = \lambda(d_n \mid R) \cdot r(d_n).
$$

The policy-aware estimator $\hat{\Delta}_{aware}$ (Eq. 5.13) also corrects for the examination bias, but because its propensity scores are based on the policy instead of the individual rankings (Eq. 5.12), it uses the same score for both rankings:

$$
\frac{\pi(\bar{R}_1 \mid q) \cdot P(o(d_n) = 1 \mid q, r, \bar{R}_1) + \pi(\bar{R}_2 \mid q) \cdot P(o(d_n) = 1 \mid q, r, \bar{R}_2)}{\pi(\bar{R}_1 \mid q) \cdot P(o(d_n) = 1 \mid q, r, \bar{R}_1) + \pi(\bar{R}_2 \mid q) \cdot P(o(d_n) = 1 \mid q, r, \bar{R}_2)} \tag{5.19}
$$
$$
\cdot \lambda(d_n \mid R) \cdot r(d_n) = \lambda(d_n \mid R) \cdot r(d_n).
$$

Then, we consider a different relevant item $d_m$ with $r(d_m) = r(d_n)$ that unlike the previous situation only appears in the top-$k$ of $\bar{R}_1$. Thus it only has a positive observance probability in $\bar{R}_1$: $P(o(d_m) = 1 \mid q, r, \bar{R}_1) > 0$ and $P(o(d_m) = 1 \mid q, r, \bar{R}_2) = 0$. Consequently, no clicks will ever be received in $\bar{R}_2$, i.e., $\bar{R} = \bar{R}_2 \rightarrow c(d_m) = 0$, thus

the expectation for $d_m$ only has to consider $\bar{R}_1$:

$$
\mathbb{E}_{o,\bar{R}}\Big[\frac{c(d_m) \cdot \lambda\big(d_m \mid R\big)}{\rho\big(o(d_m) = 1 \mid q, d_m, \bar{R}, \pi\big)}\Big]
$$
$$
= \frac{\pi(\bar{R}_1 \mid q) \cdot P\big(o(d_m) = 1 \mid q, r, \bar{R}_1\big)}{\rho\big(o(d_m) = 1 \mid q, d_m, \bar{R}_1, \pi\big)} \cdot \lambda\big(d_m \mid R\big) \cdot r(d_m).
\tag{5.20}
$$

In this situation, Condition 5.10 is not satisfied, and correspondingly, the policy-oblivious estimator (Eq. 5.8) does not give an unbiased estimate:

$$
\frac{\pi(\bar{R}_1 \mid q) \cdot P\big(o(d_m) = 1 \mid q, r, \bar{R}_1\big)}{P\big(o(d_m) = 1 \mid q, r, \bar{R}_1\big)} \cdot \lambda\big(d_m \mid R\big) \cdot r(d_m) < \lambda\big(d_m \mid R\big) \cdot r(d_m).
$$
$$
\tag{5.21}
$$

Since the policy-oblivious estimator $\hat{\Delta}_{oblivious}$ only corrects for the observance probability per displayed ranking, it is unable to correct for the zero probability in $R_2$ as no clicks on $d_m$ can occur here. As a result, the estimate is affected by the logging policy $\pi$: the more item-selection bias $\pi$ introduces (determined by $\pi(\bar{R}_1 \mid q)$) the further the estimate will deviate. Consequently, in expectation $\hat{\Delta}_{oblivious}$ will biasedly estimate that $d_n$ should be ranked higher than $d_m$, which is incorrect since both items are actually equally relevant.

In contrast, the policy-aware estimator $\hat{\Delta}_{aware}$ (Eq. 5.13) avoids this issue because its propensities are based on the logging policy $\pi$. When calculating the probability of observance conditioned on $\pi$, $P\big(o(d_m) = 1 \mid q, r, \pi\big)$ (Eq. 5.12), it takes into account that there is a $\pi(\bar{R}_2 \mid q)$ chance that $d_m$ is not displayed to the user:

$$
\frac{\pi(\bar{R}_1 \mid q) \cdot P\big(o(d_m) = 1 \mid q, r, \bar{R}_1\big)}{\pi(\bar{R}_1 \mid q) \cdot P\big(o(d_m) = 1 \mid q, r, \bar{R}_1\big)} \cdot \lambda\big(d_m \mid R\big) \cdot r(d_m) = \lambda\big(d_m \mid R\big) \cdot r(d_m). \tag{5.22}
$$

Since in this situation Condition 5.16 is true (and therefore also Condition 5.15), we know beforehand that in expectation the policy-aware estimator is unaffected by position and item-selection bias.

This concludes our illustrative example. It was meant to contrast the behavior of the policy-aware and policy-oblivious estimators in two different situations. When there is no item-selection bias, i.e., an item is displayed in the top-$k$ of all rankings the logging policy may display, both estimators provide unbiased estimates albeit using different propensity scores. However, when there is item-selection bias. i.e., an item is not always present in the top-$k$, the policy-oblivious estimator $\hat{\Delta}_{oblivious}$ no longer provides an unbiased estimate, while the policy-aware estimator $\hat{\Delta}_{aware}$ is still unbiased w.r.t. both position bias and item-selection bias.

## 5.4 Learning for Top-$k$ Metrics

This section details how counterfactual LTR can be used to optimize top-$k$ metrics, since these are the relevant metrics in top-$k$ rankings.

### 5.4.1 Top-$k$ metrics

Since top-$k$ rankings only display the $k$ highest ranked items to the user, the performance of a ranker in this setting is only determined by those items. Correspondingly, only top-$k$ metrics matter here, where items beyond rank $k$ have no effect:

$$\lambda^{metric@k}(d \mid R) = \begin{cases} \lambda^{metric}(d \mid R), & \text{if } \text{rank}(d \mid R) \leq k, \\ 0, & \text{if } \text{rank}(d \mid R) > k. \end{cases} \quad (5.23)$$

These metrics are commonly used in LTR since, usually, performance gains in the top of a ranking are the most important for the user experience. For instance, NDCG@$k$, which is the normalized version of DCG@$k$, is often used:

$$\lambda^{DCG@k}(d \mid R) = \begin{cases} -\log_2\left(1 + \text{rank}(d \mid R)\right)^{-1}, & \text{if } \text{rank}(d \mid R) \leq k, \\ 0, & \text{if } \text{rank}(d \mid R) > k. \end{cases} \quad (5.24)$$

Generally in LTR, DCG is optimized in order to maximize NDCG [13, 129]. In unbiased LTR it is not trivial to estimate the normalization factor for NDCG, further motivating the optimization of DCG instead of NDCG [2, 16].

Importantly, top-$k$ metrics bring two main challenges for LTR. First, the rank function is not differentiable, a problem for almost every LTR metric [75, 129]. Second, changes in a ranking beyond position $k$ do not affect the metric's value thus resulting in zero-gradients. The first problem has been addressed in existing LTR methods, we will now propose adaptations of these methods that address the second issue as well.

### 5.4.2 Monotonic upper bounding

A common approach for enabling optimization of ranking methods, is by finding lower or upper bounds that can be minimized or maximized, respectively. For instance, similar to a hinge loss, the rank function can be upper bounded by a maximum over score differences [54, 58]. Let $s$ be the scoring function used to rank (in descending order), then:

$$\text{rank}(d \mid R) \leq \sum_{d' \in R} \max\left(1 - \left(s(d) - s(d')\right), 0\right). \quad (5.25)$$

Alternatively, the logistic function is also a popular choice [129]:

$$\text{rank}(d \mid R) \leq \sum_{d' \in R} \log_2\left(1 + e^{s(d') - s(d)}\right). \quad (5.26)$$

Minimizing one of these differentiable upper bounds will directly minimize an upper bound on the ARP metric (Eq. 5.3).

Furthermore, Agarwal et al. [2] showed that this approach can be extended to any metric based on a monotonically decreasing function. For instance, if $\overline{\text{rank}}(d \mid R)$ is an upper bound on the $\text{rank}(d \mid R)$ function, then the following is an upper bound on the DCG loss (Eq. 5.4):

$$\lambda^{DCG}(d \mid R) \leq -\log_2\left(1 + \overline{\text{rank}}(d \mid R)\right)^{-1} = \hat{\lambda}^{DCG}(d \mid R). \quad (5.27)$$

More generally, let $\alpha$ be a monotonically decreasing function. A loss based on $\alpha$ is always upper bounded by:

$$\lambda^{\alpha}(d \mid R) = -\alpha\big(\mathrm{rank}(d \mid R)\big) \leq -\alpha\big(\overline{\mathrm{rank}}(d \mid R)\big) = \hat{\lambda}^{\alpha}(d \mid R). \qquad (5.28)$$

Though appropriate for many standard ranking metrics, $\hat{\lambda}^{\alpha}$ is not an upper bound for top-$k$ metric losses. To understand this, consider that an item beyond rank $k$ may still receive a negative score from $\hat{\lambda}^{\alpha}$, for instance, for the DCG upper bound: $\mathrm{rank}\big(d \mid R\big) > k \rightarrow \hat{\lambda}^{DCG}\big(d \mid R\big) < 0$. As a result, this is not an upper bound for a DCG@$k$ based loss.

We propose a modification of the $\hat{\lambda}^{\alpha}$ function to provide an upper bound for top-$k$ metric losses, by simply giving a positive penalty to items beyond rank $k$:

$$\hat{\lambda}^{\alpha@k}\big(d \mid R\big) = -\alpha\big(\overline{\mathrm{rank}}(d \mid R)\big) + \mathbb{1}\big[\mathrm{rank}(d \mid R) > k\big] \cdot \alpha(k). \qquad (5.29)$$

The resulting function is an upper bound on top-$k$ metric losses based on a monotonic function: $\lambda^{\alpha@k}\big(d \mid R\big) \leq \hat{\lambda}^{\alpha@k}\big(d \mid R\big)$. The main difference with $\hat{\lambda}^{\alpha}$ is that items beyond rank $k$ acquire a positive score from $\lambda^{\alpha@k}$, thus providing an upper bound on the actual metric loss. Interestingly, the gradient of $\hat{\lambda}^{\alpha@k}$ w.r.t. the scoring function $s$ is the same as that of $\hat{\lambda}^{\alpha}$.[1] Therefore, the gradient of either function optimizes an upper bound on $\lambda^{\alpha@k}$ top-$k$ metric losses, while only $\hat{\lambda}^{\alpha@k}$ provides an actual upper bound.

While this monotonic function-based approach is simple, it is unclear how coarse these upper bounds are. In particular, some upper bounds on the rank function (e.g., Eq. 5.25) can provide gross overestimations. As a result, these upper bounds on ranking metric losses may be very far removed from their actual values.

### 5.4.3 Lambda-based losses for counterfactual top-$k$ LTR

Many supervised LTR approaches, such as the well-known LambdaRank and subsequent LambdaMART methods [13], are based on Expectation Maximization (EM) procedures [28]. Recently, Wang et al. [129] introduced the LambdaLoss framework, which provides a theoretical way to prove that a method optimizes a lower bound on a ranking metric. Subsequently, it was used to prove that LambdaMART optimizes such a bound on DCG, similarly it was also used to introduce the novel LambdaLoss method which provides an even tighter bound on DCG. In this section, we will show that the LambdaLoss framework can be used to find proven bounds on counterfactual LTR losses and top-$k$ metrics. Since LambdaLoss is considered state-of-the-art in supervised LTR, making its framework applicable to counterfactual LTR could potentially provide competitive performance. Additionally, adapting the LambdaLoss framework to top-$k$ metrics further expands its applicability.

The LambdaLoss framework and its EM-optimization approach work for metrics that can be expressed in item-based gains, $G(d_n \mid q, r)$, and discounts based on position, $D\big(\mathrm{rank}(d_n \mid R)\big)$; for brevity we use the shorter $G_n$ and $D_n$, respectively, resulting in:

$$\Delta\big(R \mid q, r\big) = \sum_{d_n \in R} G(d_n \mid q, r) \cdot D\big(\mathrm{rank}(d_n \mid R)\big) = \sum_{n=1}^{|R|} G_n \cdot D_n. \qquad (5.30)$$

---

[1]We consider the indicator function to never have a non-zero gradient.

For simplicity of notation, we choose indexes so that: $n = \text{rank}(d_n \mid R)$, thus $D_n$ is always the discount for the rank $n$. Then, we differ from the existing LambdaLoss framework by allowing the discounts to be zero ($\forall n\, D_n \geq 0$), thus also accounting for top-$k$ metrics. Furthermore, items at the first rank are not discounted or the metric can be scaled so that $D_1 = 1$. Additionally, higher ranked items should be discounted less or equally: $n > m \to D_n \leq D_m$. Most ranking metrics meet these criteria; for instance, $G_n$ and $D_n$ can be chosen to match ARP or DCG. Importantly, our adaption also allows $\Delta$ to match top-$k$ metrics such as DCG@$k$ or Precision@$k$.

In order to apply the LambdaLoss framework to counterfactual LTR, we consider a general inverse-propensity-scored estimator:

$$\hat{\Delta}_{IPS}(R \mid q, c, \cdot) = \sum_{d_n : c(d_n)=1} \frac{\lambda(d_n \mid R)}{\rho\big(o(d_n) = 1 \mid q, r, \bar{R}, \pi\big)}, \tag{5.31}$$

where the propensity function $\rho$ can match either the policy-oblivious (Eq. 5.8) or the policy-aware (Eq. 5.13) estimator. By choosing

$$G_n = \frac{1}{\rho\big(o(d_n) = 1 \mid q, r, \bar{R}, \pi\big)} \text{ and } D_n = \lambda(d_n \mid R), \tag{5.32}$$

the estimator can be described in terms of gains and discounts. In contrast, in the existing LambdaLoss framework [129] gains are based on item relevance. For counterfactual top-$k$ LTR, we have designed Eq. 5.32 so that gains are based on the propensity scores of observed clicks, and the discounts can have zero values.

The EM-optimization procedure alternates between an expectation step and a maximization step. In our case, the expectation step sets the discount values $D_n$ according to the current ranking $R$ of the scoring function $s$. Then the maximization step updates $s$ to optimize the ranking model. Following the LambdaLoss framework [129], we derive a slightly different loss. With the delta function:

$$\delta_{nm} = D_{|n-m|} - D_{|n-m|+1}, \tag{5.33}$$

our differentiable *counterfactual loss* becomes:

$$\sum_{G_n > G_m} -\log_2\left(\left(\frac{1}{1 + e^{s(d_m) - s(d_n)}}\right)^{\delta_{nm} \cdot |G_n - G_m|}\right). \tag{5.34}$$

The changes we made do not change the validity of the proof provided in the original LambdaLoss paper [129]. Therefore, the counterfactual loss (Eq. 5.34) can be proven to optimize a lower bound on counterfactual estimates of top-$k$ metrics.

Finally, in the same way the LambdaLoss framework can also be used to derive counterfactual variants of other supervised LTR losses/methods such as LambdaRank or LamdbaMART. Unlike previous work that also attempted to find a counterfactual lambda-based method by introducing a pairwise-based estimator [46], our approach is compatible with the prevalent counterfactual approach since it uses the same estimator based on single-document propensities [3–5, 58, 127, 128]. Our approach suggests that the divide between supervised and counterfactual LTR methods may disappear in the future, as a state-of-the-art supervised LTR method can now be applied to the state-of-the-art counterfactual LTR estimators.

### 5.4.4 Unbiased loss selection

So far we have introduced two counterfactual LTR approaches that are proven to optimize lower bounds on top-$k$ metrics: with monotonic functions (Section 5.4.2) and through the LambdaLoss framework (Section 5.4.3). To the best of our knowledge, we are the first to introduce theoretically proven lower bounds for top-$k$ LTR metrics. Nevertheless, previous work has also attempted to optimize top-$k$ metrics, albeit through heuristic methods. Notably, Wang et al. [129] used a truncated version of the LambdaLoss loss to optimize DCG@$k$. Their loss uses the discounts $D_n$ based on full-ranking DCG but ignores item pairs outside of the top-$k$:

$$\sum_{G_n > G_m} -\mathbb{1}\big[n \leq k \vee m \leq k\big] \cdot \log_2 \left( \left( \frac{1}{1 + e^{s(d_m)-s(d_n)}} \right)^{\delta_{nm} \cdot |G_n - G_m|} \right). \quad (5.35)$$

While empirical results motivate its usage, there is no known theoretical justification for this loss, and thus it is considered a heuristic.

This leaves us with a choice between two theoretically-motivated counterfactual LTR approaches for optimizing top-$k$ metrics (Eq. 5.29 and 5.34) and an empirically-motivated heuristic (Eq. 5.35). We propose a pragmatic solution by recognizing that counterfactual estimators can unbiasedly evaluate top-$k$ metrics. Therefore, in practice one can optimize several ranking models using various approaches, and subsequently, estimate which resulting model provides the best performance. Thus, using counterfactual evaluation to select from resulting models is an unbiased method to choose between the available counterfactual LTR approaches.

## 5.5 Experimental Setup

We follow the standard setup in unbiased LTR [5, 16, 50, 58] and perform semi-synthetic experiments: queries and items are based on datasets of commercial search engines and interactions are simulated using probabilistic click models.

### 5.5.1 Datasets

We use the queries and documents from two of the largest publicly available LTR datasets: MLSR-WEB30K [95] and Yahoo! Webscope [17]. Each was created by a commercial search engine and contains a set of queries with corresponding preselected document sets. Query-document pairs are represented by feature vectors and five-grade relevance annotations ranging from not relevant (0) to perfectly relevant (4). In order to binarize the relevancy, we only consider the two highest relevance grades as relevant. The MSLR dataset contains 30,000 queries with on average 125 preselected documents per query, and encodes query-document pairs in 136 features. The Yahoo dataset has 29,921 queries and on average 24 documents per query encoded in 700 features. Presumably, learning from top-$k$ feedback is harder as $k$ becomes a smaller percentage of the number of items. Thus, we expect the MSLR dataset with more documents per query to pose a more difficult problem.

## 5.5.2   Simulating top-$k$ settings

The setting we simulate is one where interactions are gathered using a non-optimal but decent production ranker. We follow existing work [5, 50, 58] and use supervised optimization for the ARP metric on 1% of the training data. The resulting model simulates a real-world production ranker since it is much better than a random initialization but leaves enough room for improvement [58].

We then simulate user-issued queries by uniformly sampling from the training partition of the dataset. Subsequently, for each query the production ranker ranks the documents preselected by the dataset. Depending on the experimental run that we consider, randomization is performed on the resulting rankings. In order for the policy-aware estimator to be unbiased, every relevant document needs a chance of appearing in the top-$k$ (Condition 5.16). Since in a realistic setting relevancy is unknown, we choose to give every document a non-zero probability of appearing in the top-$k$. Our randomization policy takes the ranking of the production ranker and leaves the first $k-1$ documents unchanged but the document at position $k$ is selected by sampling uniformly from the remaining documents. The result is a minimally invasive randomized top-$k$ ranking since most of the ranking is unchanged and the placement of the sampled documents is limited to the least important position.

We note that many other logging policies could be applied (see Condition 5.16), e.g., an alternative policy could insert sampled documents at random ranks for less obvious randomization. Unfortunately, a full exploration of the effect of using different logging policies is beyond the scope of this chapter.

Clicks are simulated on the resulting ranking $\bar{R}$ according to position bias and document relevance. Top-$k$ position bias is modelled through the probability of observance, as follows:

$$P\big(o(d) = 1 \mid q, r, \bar{R}\big) = \begin{cases} \text{rank}(d \mid \bar{R})^{-1}, & \text{if } \text{rank}(d \mid \bar{R}) \leq k, \\ 0, & \text{if } \text{rank}(d \mid \bar{R}) > k. \end{cases} \tag{5.36}$$

The randomization policy results in the following examination probabilities w.r.t. the logging policy (cf. Eq. 5.12):

$$\begin{aligned} P\big(o(d) &= 1 \mid q, r, \pi\big) \\ &= \begin{cases} \text{rank}(d \mid \bar{R})^{-1}, & \text{if } \text{rank}(d \mid \bar{R}) < k, \\ \big(\text{rank}(d \mid \bar{R}) \cdot (|\bar{R}| - k + 1)\big)^{-1}, & \text{if } \text{rank}(d \mid \bar{R}) \geq k. \end{cases} \end{aligned} \tag{5.37}$$

The probability of a click is conditioned on the relevance of the document according to the dataset:

$$P\big(c(d) = 1 \mid q, r, \bar{R}, o\big) = \begin{cases} 1, & \text{if } r(d) = 1 \wedge o(d) = 1, \\ 0.1, & \text{if } r(d) = 0 \wedge o(d) = 1, \\ 0, & \text{if } o(d) = 0. \end{cases} \tag{5.38}$$

Note that our previous assumption that clicks only take place on relevant items (Section 5.2.2) is not true in our experiments.

Optimization is performed on training clicks simulated on the training partition of the dataset. Hyperparameter tuning is done by estimating performance on (unclipped) validation clicks simulated on the validation partition; the number of validation clicks is always 15% of the number of training clicks. Lastly, evaluation metrics are calculated on the test partition using the dataset labels.

### 5.5.3  Experimental runs

In order to evaluate the performance of the policy-aware estimator (Eq. 5.13) and the effect of item-selection bias, we compare with the following baselines: (i) The policy-oblivious estimator (Eq. 5.8). In our setting, where the examination probabilities are known beforehand, the policy-oblivious estimator also represents methods that jointly estimate these probabilities while performing LTR, i.e., the following methods reduce to this estimator if the examination probabilities are given: [3, 5, 58, 127]. (ii) A rerank estimator, an adaption of the policy-oblivious estimator. During optimization the rerank estimator applies the policy-oblivious estimator but limits the document set of an interaction $i$ to the $k$ displayed items $R_i = \{d \mid \mathrm{rank}(d|\bar{R}_i) \leq k\}$ (cf. Eq. 5.8). Thus, it is optimized to rerank the top-$k$ of the production ranker only, but during inference it is applied to the entire document set. (iii) Additionally, we evaluate performance without any cutoff $k$ or randomization; in these circumstances all three estimators (Policy-Aware, Policy-Oblivious, Rerank) are equivalent. (iv) Lastly, we use supervised LTR on the dataset labels to get a *full-information skyline*, which shows the hypothetical optimal performance.

To evaluate the effectiveness of our proposed loss functions for optimizing top-$k$ metrics, we apply the monotonic lower bound (Eq. 5.29) with a linear (Eq. 5.25) and a logistic upper bound (Eq. 5.26). Additionally, we apply several versions of the LamdbaLoss loss function (Eq. 5.34): one that optimizes full DCG, another that optimizes DCG@5, and the heuristic truncated loss also optimizing DCG@5 (Eq. 5.35). Lastly, we apply unbiased loss selection where we select the best-performing model based on the estimated performance on the (unclipped) validation clicks.

Optimization is done with stochastic gradient descent; to maximize computational efficiency we rewrite the loss (Eq. 5.5) for a propensity scoring function $\rho$ in the following manner:

$$
\begin{aligned}
\hat{\mathcal{L}} &= \frac{1}{N} \sum_{i=1}^{N} \hat{\Delta}\big(R_i | q_i, \bar{R}_i, \pi, c_i\big) \\
&= \frac{1}{N} \sum_{i=1}^{N} \sum_{d:c_i(d)=1} \frac{\lambda\big(d \mid R_i\big)}{\rho\big(o_i(d) = 1 | q_i, r, \cdot\big)} \\
&= \frac{1}{N} \sum_{q \in \mathcal{Q}} \sum_{d \in R_q} \left( \sum_{i=1}^{N} \frac{\mathbb{1}[q_i = q] \cdot c_i(d)}{\rho\big(o_i(d) = 1 \mid q, r, \cdot\big)} \right) \cdot \lambda\big(d \mid R_q\big) \\
&= \frac{1}{N} \sum_{q \in \mathcal{Q}} \sum_{d \in R_q} \omega_d \cdot \lambda\big(d \mid R_q\big).
\end{aligned}
\tag{5.39}
$$

After precomputing the document weights $\omega_d$, the complexity of computing the loss is only determined by the dataset size. This allows us to optimize over very large numbers of clicks with very limited increases in computational costs.

We optimize linear models, but our approach can be applied to any differentiable model [2]. Propensity clipping [58] is applied to training clicks and never applied to the validation clicks; we also use self-normalization [116].

## 5.6 Results and Discussion

In this section we discuss the results of our experiments and evaluate our policy-aware estimator and the methods for top-$k$ LTR metric optimization empirically.

### 5.6.1 Learning under item-selection bias

First we consider the question:

**RQ5.1** Is the policy-aware estimator effective for unbiased counterfactual LTR from top-$k$ feedback?

Figure 5.1 displays the performance of different approaches after optimization on $10^8$ clicks under varying values for $k$. Both the policy-oblivious and rerank estimators are greatly affected by the item-selection bias introduced by the cutoff at $k$. On the MSLR dataset neither approach is able to get close to optimal ARP performance, optimal DCG@5 is only reached when $k > 50$. On the Yahoo dataset, the policy-oblivous approach can only approximate optimal ARP when $k > 60$; for DCG@5 it requires $k > 25$. The rerank approach reaches optimal ARP when $k > 50$ and optimal DCG@5 when $k > 20$. Considering that on average a query in the Yahoo dataset only has 24 preselected documents, it appears that even a little item-selection bias has a substantial effect on both estimators. Furthermore, randomization appears to have a very limited positive effect on the policy-oblivious and rerank approaches. The one exception is the policy-oblivious approach when $k = 1$ where it reaches optimal performance under randomization. Here, the randomization policy gives every item an equal probability of being presented, thus trivially removing item-selection bias; additionally, there is no position bias as there is only a single position. However, besides this trivial exception, the baseline estimators are strongly affected by item-selection bias and simply logging with randomization is unable to remove the effect of item-selection bias.

In contrast, the policy-aware approach is hardly affected by the choice of $k$. It consistently approximates optimal performance in terms of ARP and DCG@5 on both datasets. On the MSLR dataset, the policy-aware approach provides near optimal ARP performance; however, for $k > 15$ there is a small but noticeable gap. We suspect that this is the result of variance from click-noise and can be closed by gathering more clicks. Across all settings, the policy-aware approach appears unaffected by the choice of $k$ and thus the effect of item-selection bias. Moreover, it consistently provides performance at least as good as the baselines; and on the Yahoo dataset it outperforms them for $k < 20$ and on the MSLR dataset outperforms them for all tested values of $k$. We note that the randomization policy is the same for all methods; in other words, under

randomization the clicks for the policy-oblivious, policy-aware and rerank approaches are acquired in the exact same way. Thus, our results show that in order to benefit from randomization, a counterfactual LTR method has to take its effect into account, hence only the policy-aware approach has improved performance.

Figure 5.2 displays the performance when learning from top-5 feedback while varying the number of clicks. Here we see that the policy-oblivious approach performance is stable after $10^5$ clicks have been gathered. The rerank approach has stable performance after $10^6$ clicks when optimized for ARP and $10^5$ for DCG@5. Both baseline approaches show biased behavior where adding additional data does not lead to improved performance. This confirms that their estimators are unable to deal with item-selection bias. In contrast, the policy-aware approach reaches optimal performance in all settings. However, it appears that the policy-aware approach requires more clicks than the no-cutoff baseline; we suspect that this difference is due to variance added by the randomization and smaller propensity scores.

In conclusion, we answer our **RQ5.1** positively: our results show that the policy-aware approach is unbiased w.r.t. item-selection bias and position bias. Where all baseline approaches are affected by item-selection bias even in small amounts, the policy-aware approach approximates optimal performance regardless of the cutoff value $k$.

### 5.6.2 Optimizing top-$k$ metrics

Next, we consider the question:

**RQ5.2** Are our novel counterfactual LTR loss functions effective for top-$k$ LTR metric optimization?

Figure 5.3 shows the performance of the policy-aware approach after optimizing different loss functions under top-5 feedback. While on the Yahoo dataset small differences are observed, on the MSLR dataset substantial differences are found. Interestingly, there seems to be no advantage in optimizing for DCG@5 instead of full DCG with the LambdaLoss. Furthermore, the monotonic loss function works very well with a linear upper bound, yet poorly when using the log upper bound. On both datasets the heuristic truncated LambdaLoss loss function provides the best performance, despite being the only method without a theoretical basis. When few clicks are available, the differences change; e.g., the monotonic loss function with a log upper bound outperforms the other losses on the MSLR dataset when fewer than $10^5$ clicks are available.

Finally, we consider unbiased loss selection; Figure 5.3 displays both the performance of the selected models and the estimated performance on which the selections are based. For the most part the optimal models are selected, but variance does cause mistakes in selection when few clicks are available. Thus, unbiased optimal loss selection seems effective as long as enough clicks are available.

In conclusion, we answer **RQ5.2** positively: our results indicate that the truncated counterfactual LambdaLoss loss function is most effective at optimizing DCG@5. Using this loss, our counterfactual LTR method reaches state-of-the-art performance comparable to supervised LTR on both datasets. Alternatively, our proposed unbiased

loss selection method can choose optimally between models that are optimized by different loss functions.

## 5.7  Related Work

Section 5.2.1 has discussed supervised LTR and Section 5.2.2 has described the existing counterfactual LTR framework; this section contrasts additional related work with our policy-aware approach.

Interestingly, some existing work in unbiased LTR was performed in top-$k$ rankings settings [3, 4, 127, 128]. Our findings suggest that the results of that work are affected by item-selection bias and that there is the potential for considerable improvements by applying the policy-aware method.

Carterette and Chandar [16] recognized that counterfactual evaluation cannot evaluate rankers that retrieve items that are unseen in the interaction logs, essentially due to a form of item-selection bias. Their proposed solution is to gather new interactions on rankings where previously unseen items are randomly injected. Accordingly, they adapt propensity scoring to account for the random injection strategy. In retrospect, this approach can be seen as a specific instance of our policy-aware approach. In contrast, we have focused on settings where item-selection bias takes place systematically and propose that logs should be gathered by any policy that meets Condition 5.16. Instead of expanding the logs to correct for missing items, our approach avoids systematic item-selection bias altogether.

Other previous work has also used propensity scores based on a logging policy and examination probabilities. Komiyama et al. [67] and subsequently Lagrée et al. [69] use such propensities to find the optimal ranking for a single query by casting the ranking problem as a multiple-play bandit. Li et al. [72] use similar propensities to counterfactually evaluate ranking policies where they estimate the number of clicks a ranking policy will receive. Our policy-aware approach contrasts with these existing methods by providing an unbiased estimate of LTR-metric-based losses, and thus it can be used to optimize LTR models similar to supervised LTR.

Lastly, online LTR methods where interactive processes learn from the user [132] also make use of stochastic ranking policies. They correct for biases through randomization in rankings but do not use an explicit model of examination probabilities. To contrast with counterfactual LTR, while online LTR methods appear to provide robust performance [50], they are not proven to unbiasedly optimize LTR metrics [82, 84]. Unlike counterfactual LTR, they are not effective when applied to historical interaction logs [43].

## 5.8  Conclusion

In this chapter, we have proposed a policy-aware estimator for LTR, the first counterfactual method that is unbiased w.r.t. both position bias and item-selection bias. Our experimental results show that existing policy-oblivious approaches are greatly affected by item-selection bias, even when only small amounts are present. In contrast, the proposed policy-aware LTR method can learn from top-$k$ feedback without being affected

by the choice of $k$. Furthermore, we proposed three counterfactual LTR approaches for optimizing top-$k$ metrics: two theoretically proven lower bounds on DCG@$k$ based on monotonic functions and the LambdaLoss framework, respectively, and another heuristic truncated loss. Additionally, we introduced unbiased loss selection that can choose optimally between models optimized with different loss functions. Together, our contributions provide a method for learning from top-$k$ feedback and for top-$k$ metrics.

With these contributions, we can answer the thesis research questions **RQ4** and **RQ5** positively: with the policy-aware estimator counterfactual LTR is applicable to top-$k$ ranking settings; furthermore, we have shown that the state-of-the-art supervised LTR LambdaLoss method can be used for counterfactual LTR. To the best of our knowledge, this is the first counterfactual LTR method that is unbiased in top-$k$ ranking settings. Additionally, this chapter also serves to further bridge the gap between supervised and counterfactual LTR methods, as we have shown that state-of-the-art lambda-based supervised LTR methods can be applied to the state-of-the-art counterfactual LTR estimators. Therefore, the contributions of this chapter have greatly extended the capability of the counterfactual LTR approach and further connected it with the supervised LTR field.

Future work in supervised LTR could verify whether potential novel supervised methods can be applied to counterfactual losses. A limitation of the policy-aware LTR approach is that the logging policy needs to be known; future work could investigate whether a policy estimated from logs also suffices [72, 74]. Finally, existing work on bias in recommendation [107] has not considered position bias, thus we anticipate further opportunities for counterfactual LTR methods for top-$k$ recommendations.

The remaining chapters of this thesis will continue to build on the policy-aware estimator. Chapter 6 introduces a counterfactual LTR algorithm that uses the policy-aware estimator to combine properties of tabular models and feature-based models. Furthermore, Chapter 7 looks at how the policy-aware estimator can be used for ranker evaluation. It introduces an algorithm that optimizes the logging policy to reduce variance when using the policy-aware estimator for evaluation. Lastly, Chapter 8 introduces a novel intervention-aware estimator inspired by the policy-aware estimator. This novel estimator takes the policy-aware approach even further by considering the effect of all logging policies used during data gathering. The intervention-aware approach thus also considers the case where the logging policy is updated during the gathering of data. Besides the policy-aware estimator, Chapter 6, Chapter 7, and Chapter 8 all use the adaption of LambdaLoss for counterfactual LTR derived in this chapter.
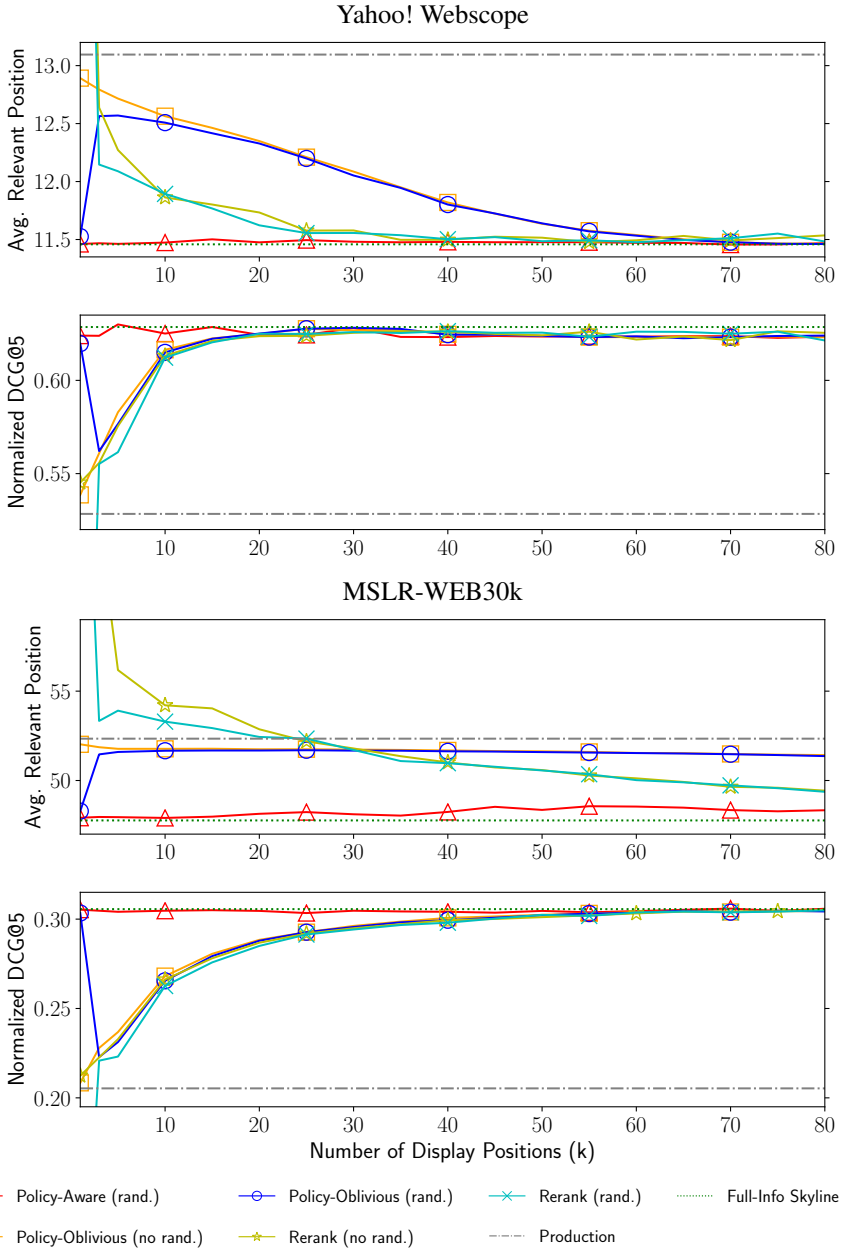
Figure 5.1: The effect of item-selection bias on different estimators. Optimization on $10^8$ clicks simulated on top-$k$ rankings with varying number of display positions ($k$), with and without randomization (for each datapoint $10^8$ clicks were simulated independently). Results on the *Yahoo* dataset and *MSLR* dataset. The top graph per dataset optimizes the average relevance position through the linear upper bound (Eq. 5.25); the bottom graph per dataset optimizes DCG@5 using the truncated LambdaLoss (Eq. 5.35).
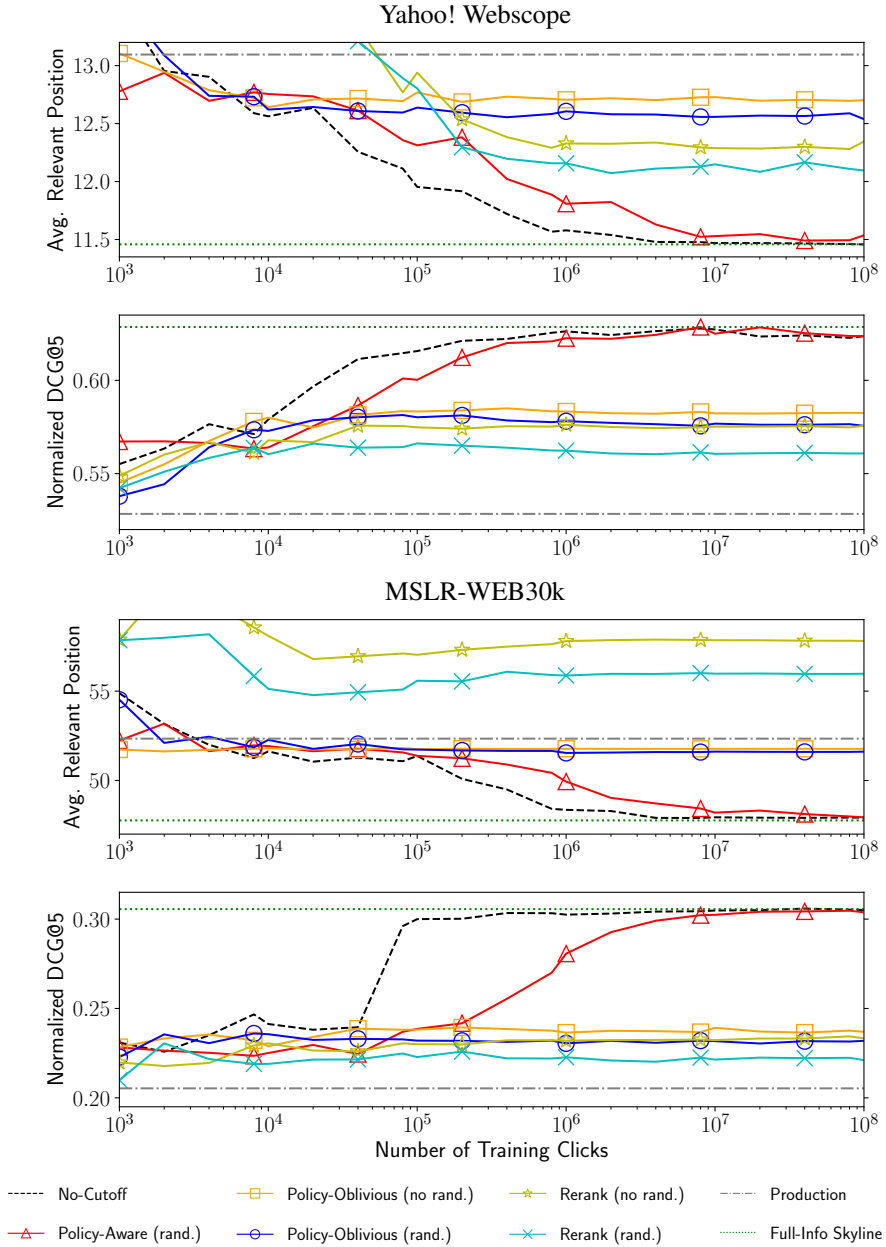
Figure 5.2: Performance of different estimators learning from different numbers of clicks simulated on top-5 rankings, with and without randomization. Results on the *Yahoo* dataset and *MSLR* dataset. The top graph per dataset optimizes the average relevance position through the linear upper bound (Eq. 5.25); the bottom graph per dataset optimizes DCG@5 using the truncated LambdaLoss (Eq. 5.35).
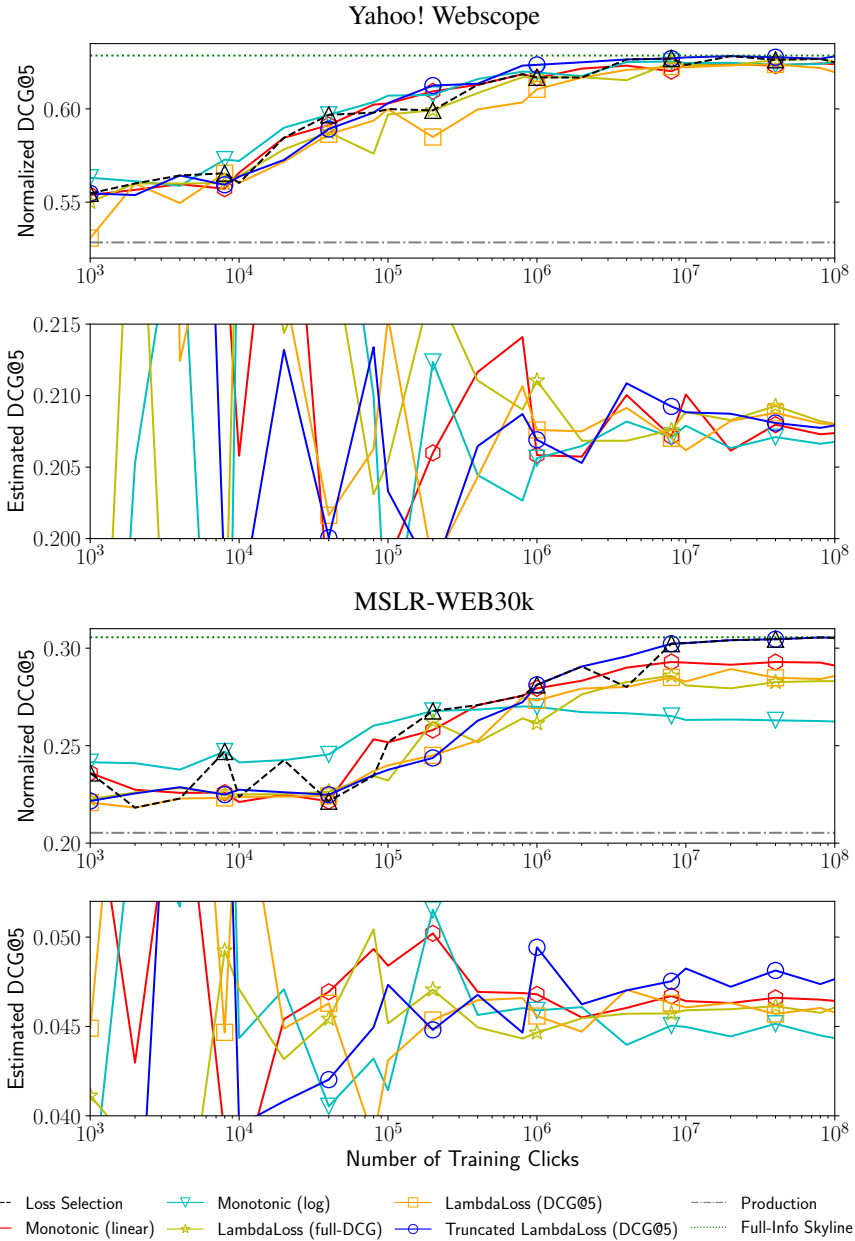
Figure 5.3: Performance of the policy-aware estimator (Eq. 5.13) optimizing DCG@5 using different loss functions. The loss selection method selects the estimated optimal model based on clicks gathered on separate validation queries. Varying numbers of clicks on top-5 rankings with randomization, the number of validation clicks is 15% of the number of training clicks.

## 5.A   Notation Reference for Chapter 5

| Notation | Description |
|---|---|
| $k$ | the number of items that can be displayed in a single ranking |
| $i$ | an iteration number |
| $Q$ | set of queries |
| $q$ | a user-issued query |
| $d$ | an item to be ranked |
| $r(d, q)$, $r(d)$ | the relevance of item $d$ w.r.t. query $q$ |
| $R$ | a ranked list |
| $\bar{R}$ | a ranked list that was displayed to the user |
| $\lambda(d \mid R)$ | a metric that weights items depending on their display rank |
| $c_i(d)$ | a function indicating item $d$ was clicked at iteration $i$ |
| $o_i(d)$ | a function indicating item $d$ was observed at iteration $i$ |
| $\pi$ | a logging policy |
| $\pi(\bar{R} \mid q)$ | the probability that policy $\pi$ displays ranking $\bar{R}$ for query $q$ |
| $\mathrm{rank}(d \mid \bar{R})$ | the rank of item $d$ in displayed ranking $\bar{R}$ |
| $\rho$ | a propensity function used to represent any IPS estimator |
| $s(d)$ | the score given to item $d$ by ranking model $s$, used to sort items by |

# 6

# Combining Generalized and Specialized Models in Counterfactual Learning to Rank

So far, this thesis has only addressed feature-based Learning to Rank (LTR) – the optimization of models that rank items based on their features – as opposed to tabular online LTR – which optimizes a ranking directly, thus not using any scoring models. A big advantage of feature-based LTR is that its model can be applied to previously unseen queries and items. As a result, it provides very robust performance in previously unseen circumstances. However, their behavior is often limited by the available features: in practice they do not provide enough information to determine the optimal ranking. In stark contrast, tabular LTR memorizes rankings, instead of using a features to predict them. Consequently, tabular LTR is not limited by which features are available and can potentially always find the optimal ranking. Despite this potential, tabular LTR does not generalize: it cannot transfer learned behavior to previously unseen queries or items. In other words, tabular LTR has the potential to specialize – perform very well in circumstances encountered often – whereas feature-based LTR is good at generalization – performing well overall, including previously unseen circumstances. In this chapter we investigate whether the advantageous properties of these two areas can be combined in the counterfactual LTR framework, and thus we address the thesis research question:

**RQ6** Can the specialization ability of tabular online LTR be combined with the robust feature-based approach of counterfactual LTR?

In this chapter we introduce a framework for Generalization and Specialization (GENSPEC) for counterfactual learning from logged bandit feedback. GENSPEC is designed for problems that can be divided in many non-overlapping contexts. It simultaneously learns a generalized policy – optimized for high performance across all contexts – and many specialized policies – each optimized for high performance in a single context. Using high-confidence bounds on the relative performance of policies, per context GENSPEC decides whether to deploy a specialized policy, the general policy, or the current logging policy. By doing so GENSPEC combines the high

---

This chapter was submitted as [87]. Appendix 6.C gives a reference for the notation used in this chapter.

performance of successfully specialized policies with the safety and robustness of a generalized policy.

While GENSPEC is applicable to many different bandit problems, we focus on query-specialization for counterfactual learning to rank, where a context consists of a query submitted by a user. Here we learn both a single general feature-based model for robust performance across queries, and many memory-based models, each of which is highly specialized for a single query, GENSPEC then chooses which model to deploy on a per query basis. Our results show that GENSPEC leads to massive performance gains on queries with sufficient click data, while still having safe and robust behavior on queries with little or noisy data.

## 6.1   Introduction

*Generalization* is an important goal for most machine learning algorithms: models should perform well across a large range of contexts, especially previously unseen contexts [10]. *Specialization*, the ability to perform well in a single context, is often disfavored over generalization because the latter is more robust [37]. Generally, the same trade-off pertains to contextual bandit problems [70, Chapter 18]. There, the goal is to find a policy that maximizes performance over the full distribution of contextual information. While a specialized policy, i.e., a policy optimized on a subset of possible contexts, could outperform a generalized policy on that subset, most likely it compromises performance on other contexts to do so since specialization comes with a risk of overfitting: applying a policy that is specialized in a specific set of contexts to different contexts [22, 37]. As a consequence, generalization is often preferred as it avoids this issue.

In this chapter, we argue that, depending on the circumstances, specialization may be preferable over generalization, specifically, if with high-confidence it can be guaranteed that a specialized policy is only deployed in contexts where it outperforms policies optimized for generalization. We focus on counterfactual learning for contextual bandit problems where contexts can be split into non-overlapping sets. We simultaneously train (i) a generalized policy that performs well across all contexts, and (ii) many specialized policies, one for each of a specific set of contexts. Thus, per context there is a choice between three policies: (i) the logging policy used to gather data, (ii) the generalized policy, and (iii) the specialized policy. Depending on the circumstances, e.g., the amount of data available, noise in the data, or the difficulty of the task, a different policy will perform best in a specific context [22]. To reliably choose between policies, we estimate high-confidence bounds [119] on the relative performance differences between policies and then choose conservatively: we only apply a specialized policy instead of the generalized policy or logging policy if the lower bounds on their differences in performance are positive in a specific context. Otherwise, the generalized policy is only applied if with high-confidence it outperforms the logging policy across all contexts. We call this approach the *Generalization and Specialization* (GENSPEC) framework: it trains both generalized and specialized policies and results in a meta-policy that chooses between them using high-confidence bounds. The GENSPEC meta-policy is particularly powerful because it can combine the properties of different models: for instance, a

generalized policy using a feature-based model can be overruled by a specialized policy using a tabular model that has memorized the best actions. GENSPEC promises the best of two worlds: the safe robustness of a generalized policy with the potentially high performance of a specialized policy.

To evaluate the GENSPEC approach, we apply it to query-specialization in the setting of *Counterfactual Learning to Rank* (LTR). Existing approaches in this field either generalize – by learning a ranking model that ranks items based on their features and generalizes well across all queries [58] – or they specialize – by learning tabular ranking models that are specific to a single query and cannot be applied to any other query [70, 138]. By viewing each query as a different context, GENSPEC learns both a generalized ranker and many specialized tabular rankers, and subsequently chooses which ranker to apply per query. Our empirical results show that GENSPEC combines the advantages of both approaches: very high performance on queries where sufficiently many interactions were observed for successful specialization, and safe robust performance on queries where interaction data is limited or noisy.

Our main contributions are:

1. an adaptation of existing counterfactual high-confidence bounds for relative performance between ranking policies;

2. the GENSPEC framework that simultaneously learns generalized and specialized ranking policies plus a meta-policy that decides which to deploy per context.

To the best of our knowledge, GENSPEC is the first counterfactual LTR method to simultaneously train generalized and specialized models, and reliably choose between them using high-confidence bounds.

## 6.2 Background: Learning to Rank

This section covers the basics of counterfactual LTR.

### 6.2.1 Supervised learning to rank

The LTR task has been approached as a contextual bandit problem before [68, 70, 117, 132]. The differentiating characteristic of the LTR task is that actions are *rankings*, thus they consist of an ordered set of $K$ items: $a = (d_1, d_2, \ldots, d_K)$. The contextual information often contains a user-issued search query, features based on the items available for ranking and item-query combinations, information about the user, among other miscellaneous information. Since our focus is query specialization, we record the query separately; thus, at each time step $i$ contextual information $x_i$ and a single query $q_i \in \{1, 2, 3, \ldots\}$ are active: $x_i, q_i \sim P(x, q)$. Let $\Delta$ indicate the reward for a ranking $a$. A policy $\pi$ should maximize the expected reward [58, 75]:

$$\mathcal{R}(\pi) = \iint \Big( \sum_{a \in \pi} \Delta(a \,|\, x, q, r) \cdot \pi(a \,|\, x, q) \Big) P(x, q) \, dx \, dq. \tag{6.1}$$

Commonly, in LTR the reward for a ranking $a$ is a linear combination of the relevance scores of the items in $a$, weighted according to their rank. We use $r(d \mid x, q)$ to denote the relevance score of item $d$ and $\lambda\big(rank(d \mid a)\big)$ for the weight per rank, resulting in:

$$\Delta(a \mid x, q, r) = \sum_{d \in a} \lambda\big(rank(d \mid a)\big) \cdot r(d \mid x, q). \tag{6.2}$$

A common choice is to optimize the *Discounted Cumulative Gain* (DCG) metric; $\lambda$ can be chosen accordingly:

$$\lambda^{DCG}\big(rank(d \mid a)\big) = \log_2\big(1 + rank(d \mid a)\big)^{-1}. \tag{6.3}$$

When the relevance function $r$ is given, maximizing $\mathcal{R}$ can be done through traditional LTR in a supervised manner [13, 75, 129].

## 6.2.2 Counterfactual learning to rank

In practice, the relevance score $r$ is often unknown or requires expensive annotation [17, 27, 95, 104]. An attractive alternative comes from LTR based on historical interaction logs, which takes a counterfactual approach [58, 127]. Let $\pi_0$ be the logging policy that was used when interactions were logged:

$$a_i \sim \pi_0(a \mid x_i, q_i). \tag{6.4}$$

LTR focusses mainly on clicks in interactions; clicks are strongly affected by *position bias* [25]. This bias arises because users often do not examine all items presented to them, and only click on examined items. As a result, items that are displayed in positions that are more often examined are also more likely to be clicked, without necessarily being more relevant. Let $o_i(d) \in \{0, 1\}$ indicate whether item $d$ was examined by the user or not:

$$o_i(d) \sim P\big(o(d) \mid a_i\big). \tag{6.5}$$

We use $c_i(d) \in \{0, 1\}$ to indicate whether $d$ was clicked at time step $i$:

$$c_i(d) \sim P\big(c(d) \mid o_i(d), r(d \mid x, q)\big). \tag{6.6}$$

We assume that click probabilities are only dependent on whether an item was examined, $o_i(d)$, and its relevance, $r(d \mid x, q)$. Furthermore, we make the common assumption that clicks only occur on examined items [58, 127], thus:

$$P\big(c(d) = 1 \mid o(d) = 0, r(d \mid x, q)\big) = 0. \tag{6.7}$$

Moreover, we assume that, given examination, more relevant documents are more likely to be clicked. Specifically, click probability is proportional to relevance with an offset $\mu \in \mathbb{R}_{>0}$:

$$P\big(c(d) = 1 \mid o(d) = 1, r(d \mid x, q)\big) \propto r(d \mid x, q) + \mu. \tag{6.8}$$

The data used for counterfactual LTR consists of observed clicks $c_i$, propensity scores $\rho_i$, contextual information $x_i$ and query $q_i$ for $N$ interactions:

$$\mathcal{D} = \big\{(c_i, a_i, \rho_i, x_i, q_i)\big\}_{i=1}^{N}. \tag{6.9}$$

We apply the policy-aware approach [86] (Chapter 5) and base $\rho$ both on the examination probability of the user and the behavior of the policy:

$$\rho_i(d) = \sum_{a \in \pi_0} P\big(o_i(d) = 1 \mid a\big) \cdot \pi_0(a \mid x_i, q_i). \tag{6.10}$$

The estimated reward based on $\mathcal{D}$ is now:

$$\hat{\mathcal{R}}(\pi \mid \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \sum_{a \in \pi} \hat{\Delta}(a \mid c_i, \rho_i) \cdot \pi(a \mid x_i, q_i), \tag{6.11}$$

where $\hat{\Delta}$ is an Inverse Propensity Scoring (IPS) estimator:

$$\hat{\Delta}(a \mid c_i, \rho_i) = \sum_{d \in a} \frac{\lambda\big(rank(d \mid a)\big) \cdot c_i(d)}{\rho_i(d)}. \tag{6.12}$$

Since the reward $r$ is not observed directly, clicks are used as implicit feedback, which is a biased and noisy indicator of relevance. The unbiased estimate $\hat{\mathcal{R}}$ can be used for unbiased evaluation and optimization since:[1]

$$\arg\max_{\pi} \mathbb{E}_{o_i, a_i}\big[\hat{\mathcal{R}}(\pi \mid \mathcal{D})\big] = \arg\max_{\pi} \mathcal{R}(\pi). \tag{6.13}$$

Previous work has introduced several methods for maximizing $\hat{\mathcal{R}}$ so as to optimize different LTR metrics [2, 58].

   This concludes our description of the counterfactual LTR basics; importantly, ranking policies can be optimized from clicks without being affected by the logging policy or the users' position bias.

## 6.3  GENSPEC for Query Specialization

This section introduces the GENSPEC framework and applies it to query specialization for LTR. Section 6.6 details how it can be applied to the general contextual bandit problem.

### 6.3.1  Generalization and query specialization

We will now propose the first part of the GENSPEC framework, which produces a general policy $\pi_g$ and, for each query $q$, a specialized policy $\pi_q$.

   GENSPEC uses the logged data $\mathcal{D}$ both to train policies and to evaluate relative performance; to avoid overfitting we split $\mathcal{D}$ in a training partition $\mathcal{D}^{train}$ and a policy-selection partition $\mathcal{D}^{sel}$ so that $\mathcal{D} = \mathcal{D}^{train} \cup \mathcal{D}^{sel}$ and $\mathcal{D}^{train} \cap \mathcal{D}^{sel} = \emptyset$.

   A policy has optimal generalization performance if it maximizes performance across *all queries*. Thus, given the generalization policy space $\Pi_g$, the optimal general policy is:

$$\pi_g = \arg\max_{\pi \in \Pi_g} \hat{\mathcal{R}}(\pi \mid \mathcal{D}^{train}). \tag{6.14}$$

---

[1]See Appendix 6.A for a proof.

Alternatively, we can also choose to optimize performance for a single query $q$. First, we only select the datapoints in $\mathcal{D}$ where query $q$ was issued:

$$\mathcal{D}_q = \big\{ (c_i, a_i, \rho_i, x_i, q_i) \in \mathcal{D} \mid q_i = q \big\}. \tag{6.15}$$

Then the policy $\pi_q$ that is specialized for query $q$ is the policy in the specialization policy space $\Pi_q$ that maximizes the performance when query $q$ is issued:

$$\pi_q = \underset{\pi \in \Pi_q}{\arg\max} \; \hat{\mathcal{R}}(\pi \mid \mathcal{D}_q^{train}). \tag{6.16}$$

The motivation for $\pi_q$ is that it has the potential to provide better performance than $\pi_g$ when $q$ is issued. We may expect $\pi_q$ to outperform $\pi_g$ because $\pi_g$ may compromise performance on the query $q$ for better performance across all queries, whereas $\pi_q$ never makes such compromises. Furthermore, $\Pi_q$ could contain more optimal policies than $\Pi_g$, because the policies in $\Pi_g$ have to be applicable to all queries whereas $\Pi_q$ can make use of specific properties of $q$. However, it is also possible that $\pi_g$ and $\pi_q$ provide the same performance. Moreover, since $\mathcal{D}_q$ is a subset of $\mathcal{D}$, the optimization of $\pi_q$ is more vulnerable to noise in the data. As a result, the true performance of $\pi_q$ for query $q$ could be worse than that of $\pi_g$, especially when $\mathcal{D}_q$ is substantially smaller than $\mathcal{D}$.

In other words, a priori it is unclear whether $\pi_g$ or $\pi_q$ are preferred. We thus need a method to estimate the optimal choice with a reasonable amount of confidence.

## 6.3.2  Safely choosing between policies

We will now propose the other part of our GENSPEC framework: a meta-policy that safely chooses between deploying $\pi_g$ and $\pi_q$ per query $q$. We wish to avoid deploying $\pi_q$ when it performs worse than $\pi_g$, and similarly, avoid deploying $\pi_g$ when it is outperformed by the logging policy $\pi_0$. Recently, a method for safe policy deployment was introduced by Jagerman et al. [51] based on high-confidence bounds [119]. The intuition behind their method is that a learned policy $\pi$ should not be deployed before we can be highly confident that it outperforms the logging policy $\pi_0$, otherwise it is safer to keep the logging policy in deployment.

While previous work has bounded the performance of individual policies [51, 119], we instead bound the *difference* in performance between two policies directly. Let $\delta(\pi_1, \pi_2)$ indicate the true difference in performance between a policy $\pi_1$ and policy $\pi_2$:

$$\delta(\pi_1, \pi_2) = \mathcal{R}(\pi_1) - \mathcal{R}(\pi_2). \tag{6.17}$$

Knowing $\delta(\pi_1, \pi_2)$ allows us to optimally choose which of the two policies to deploy. However, we can only estimate its value from historical data $\mathcal{D}$:

$$\hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) = \hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}). \tag{6.18}$$

For brevity, let $R_{i,d}$ indicate the inverse-propensity-scored difference for a single document $d$ at interaction $i$:

$$R_{i,d} = \frac{c_i(d)}{\rho_i(d)} \sum_{a \in \pi_1 \cup \pi_2} \big( \pi_1(a|x_i, q_i) - \pi_2(a|x_i, q_i) \big) \cdot \lambda\big(rank(d \mid a)\big). \tag{6.19}$$

Then, for computational efficiency we rewrite:

$$\hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \sum_{d \in a_i} R_{i,d} = \frac{1}{|\mathcal{D}|K} \sum_{(i,d) \in \mathcal{D}} K \cdot R_{i,d}. \qquad (6.20)$$

For notational purposes, we let $\sum_{(i,d) \in \mathcal{D}}$ iterate over all actions $a_i$ and $K$ documents $d$ per action $a_i$. With the confidence parameter $\epsilon \in [0, 1]$, setting $b$ to be the maximum possible absolute value for $R_{i,d}$, i.e., $b = \frac{\max \lambda(\cdot)}{\min \rho}$, and

$$\nu = \frac{2|\mathcal{D}|K \ln \left( \frac{2}{1-\epsilon} \right)}{|\mathcal{D}|K - 1} \sum_{(i,d) \in \mathcal{D}} \left( K \cdot R_{i,d} - \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) \right)^2,$$

we follow Thomas et al. [119] to get the high-confidence bound:

$$CB(\pi_1, \pi_2 \mid \mathcal{D}) = \frac{7Kb \ln \left( \frac{2}{1-\epsilon} \right)}{3(|\mathcal{D}|K - 1)} + \frac{1}{|\mathcal{D}|K} \cdot \sqrt{\nu}. \qquad (6.21)$$

In turn, this provides us with the following upper and lower confidence bounds on $\delta$:

$$LCB(\pi_1, \pi_2 \mid \mathcal{D}) = \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) - CB(\pi_1, \pi_2 \mid \mathcal{D})$$
$$UCB(\pi_1, \pi_2 \mid \mathcal{D}) = \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) + CB(\pi_1, \pi_2 \mid \mathcal{D}). \qquad (6.22)$$

As proven by Thomas et al. [119], with at least a probability of $\epsilon$ they bound the true value of $\delta(\pi_1, \pi_2)$:

$$P\Big( \delta(\pi_1, \pi_2) \in \big[ LCB(\pi_1, \pi_2 \mid \mathcal{D}), UCB(\pi_1, \pi_2 \mid \mathcal{D}) \big] \Big) > \epsilon. \qquad (6.23)$$

These guarantees allow us to safely choose between policies per query $q$. We apply a doubly conservative strategy: $\pi_g$ is not deployed before we are confident that it outperforms $\pi_0$ across all queries; and $\pi_q$ is not deployed before we are confident that it outperforms both $\pi_g$ and $\pi_0$ on query $q$. This strategy results in the GENSPEC meta-policy $\pi_{GS}$:

$$\pi_{GS}(a \mid x, q) = \begin{cases} \pi_q(a \mid x, q), & \text{if } \big( LCB(\pi_q, \pi_g \mid \mathcal{D}_q^{sel}) > 0 \\ & \qquad \wedge\ LCB(\pi_q, \pi_0 \mid \mathcal{D}_q^{sel}) > 0 \big), \\ \pi_g(a \mid x, q), & \text{if } \big( LCB(\pi_q, \pi_g \mid \mathcal{D}_q^{sel}) \leq 0 \\ & \qquad \wedge LCB(\pi_g, \pi_0 \mid \mathcal{D}^{sel}) > 0 \big), \\ \pi_0(a \mid x, q), & \text{otherwise.} \end{cases} \qquad (6.24)$$

In theory, this approach can make use of the potential gains of specialization while avoiding its risks. For instance, if the policy-selection partition $\mathcal{D}_q^{sel}$ is very small, it may be heavily affected by noise, so that the confidence bound $CB$ will be wide and $\pi_q$ will not be deployed. Simultaneously, $\mathcal{D}^{sel}$ may be large enough so that with high-confidence $\pi_g$ is deployed.

We expect that in practice the relative bounding of GENSPEC is much more data-efficient than the Safe Exploration Algorithm (SEA) approach by Jagerman et al. [51].

SEA computes an upper bound on the trusted policy and a lower bound on a learned policy, and only deploys the learned policy if its lower bound is greater than the other's upper bound. When the learned policy has higher performance than the other, we expect the relative bounds of GENSPEC to require less data to be certain about this difference than the SEA bounds. In Appendix 6.B we theoretically analyze the difference between these approaches and conclude that the relative bounding of GENSPEC is more efficient if there is a positive covariance between $\hat{\mathcal{R}}(\pi_1 \mid \mathcal{D})$ and $\hat{\mathcal{R}}(\pi_2 \mid \mathcal{D})$. Because both estimates are based on the same interaction data $\mathcal{D}$, a high covariance is extremely likely.

Previous work has described safety constraints for policy deployment [51, 62, 131]. The authors assume that a baseline policy exists whose behavior is considered safe; other policies are considered unsafe if their performance is worse than the baseline policy by a certain margin. If the logging policy is taken to be the baseline policy, then GENSPEC can meet such constraints [51]. We note that while the safety guarantee is strong for a single bound (Eq. 6.23), when applied to a large number of queries the probability of at least one incorrect bound greatly increases. This *problem of multiple comparisons* may cause some non-optimal policies to be deployed for some queries. Since we mainly care about overall performance this is not expected to be an issue; however, in cases where safety constraints are very important, $\epsilon$ can be chosen to account for the number of comparisons.

### 6.3.3   Summary of the GENSPEC method

This completes our introduction of the GENSPEC framework for query specialization. Figure 6.1 visualizes our approach to query specialization. We learn from historical interactions gathered using a logging policy $\pi_0$; the interactions are divided into a training and policy-selection partition per query. Subsequently, a policy is optimized for generalization — to perform well across all queries — and for each query a policy is optimized for specialization — to perform well for a single query. While specialization can potentially maximize performance on a specific query, it brings more risks than generalization, since a general policy is optimized on more data and may provide better performance on previously unseen queries. As a solution to this dilemma, we propose a strategy that uses high-confidence bounds on the differences in performance between policies. These bounds are then used to choose safely between the deployment of the logging, general and specialized policies. In theory, GENSPEC combines the best of both worlds: the high potential of specialization and the broad safety of generalization.

## 6.4   Experimental Setup

This section discusses our experimental setup and the policies used to evaluate the GENSPEC framework.
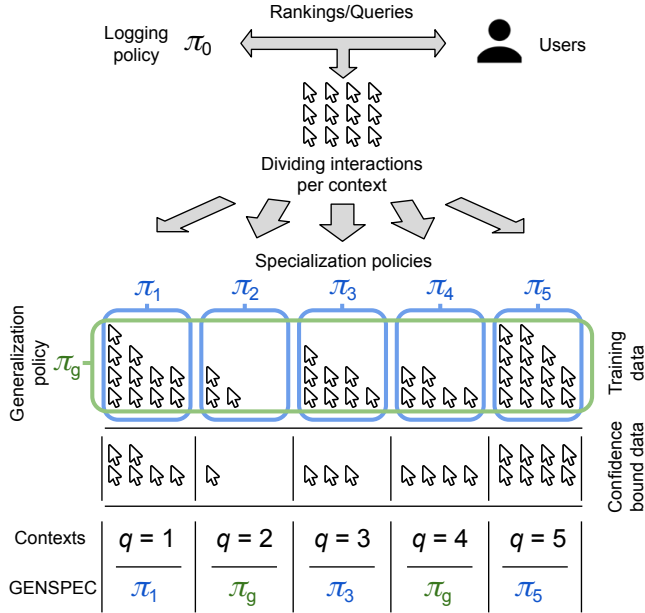
Figure 6.1: Visualization of the GENSPEC framework applied to query specialization for counterfactual LTR. The data $\mathcal{D}$ is divided per query $q$, many specialized policies $\pi_1, \pi_2, \ldots$ are each optimized for a single query $q \in \{1, 2, \ldots\}$, and single a general policy $\pi_g$ is learned on the data across all queries. Finally, GENSPEC decides which policy to deploy per context, based on high-confidence bounds.

## 6.4.1  Setup and evaluation

To evaluate the GENSPEC framework, we make use of a semi-synthetic experimental setup: queries, relevance judgements, and documents come from industry datasets, while biased and noisy user interactions are simulated using probabilistic user models. This setup is very common in the counterfactual LTR and online LTR literature [2, 58, 84]. We make use of the three largest LTR industry datasets: *Yahoo! Webscope* [17], *MSLR-WEB30k* [95], and *Istella* [27]. Each consists of a set of queries, with for each query a preselected set of documents, document-query combinations are only represented by feature vectors and a label indicating relevance according to expert annotators. Labels range from 0 (not relevant) to 4 (perfectly relevant): $r(d \mid x, q) \in \{0, 1, 2, 3, 4\}$. User issued queries are simulated by uniformly sampling from the training and validation partitions of the datasets. Displayed rankings are generated by a logging ranker using a linear model optimized on $1\%$ of the training partition using supervised LTR [58]. Then, user examination is simulated with probabilities inverse to the displayed rank of a document: $P(o(d) = 1 \mid a) = \frac{1}{rank(d \mid a)}$. Finally, user clicks are generated according to the following formula using a single parameter $\alpha \in \mathbb{R}$:

$$P\big(c(d) = 1 \mid o(d) = 1, r(d \mid x, q)\big) = 0.2 + \alpha \cdot r(d \mid x, q). \tag{6.25}$$

In our experiments, we use $\alpha = 0.2$ and $\alpha = 0.025$; the former represents a near-ideal setting where relevant documents receive a very large number of clicks, the latter represents a more noisy and harder setting where the large majority of clicks are on non-relevant documents. Clicks are only generated on the training and validation partitions, 30% of training clicks are separated for policy selection ($\mathcal{D}^{sel}$), hyperparameter optimization is done using counterfactual evaluation with clicks on the validation partition [58].

Some of our baselines are online bandit algorithms, for these baselines no clicks are separated for $\mathcal{D}^{sel}$, and the algorithms are run online: this means clicks are not gathered using the logging policy but by applying the algorithms in an online interactive setting.

The evaluation metric we use is normalized DCG (Eq. 6.3) [53] using the ground-truth labels from the datasets. Unlike most LTR work we do not apply a rank-cutoff when computing the metric, thus, an NDCG of $1.0$ indicates that *all* documents are ranked perfectly (not just the top-$k$). We separately calculate performance on the test set (Test-NDCG), to evaluate performance on previously unseen queries, and the training set (Train-NDCG). The total number of clicks is varied up to $10^9$ in total, uniformly spread over all queries, the differences in Train-NDCG when more clicks are added allows us to evaluate performance on queries with different levels of popularity.

## 6.4.2 Choice of policies

For the generalization policy space $\Pi_g$ we use feature-based ranking models. This is a natural choice as they can be applied to any query, including previously unseen ones. However, the available features could limit the possible behavior of the policies. We use linear models for $\Pi_g$; optimization is done on $\mathcal{D}^{train}$ following previous counterfactual LTR work [2]. This results in a learned scoring function $f(d, x, q) \in \mathbb{R}$ according to which items are ranked; due to score-ties there can be multiple valid rankings:

$$\mathcal{A}_g(x, q) = \{a \mid \forall(d_n, d_m) \in x, (f(d_n, x, q) > f(d_m, x, q) \to d_n \succ_a d_m)\}. \quad (6.26)$$

The general policy $\pi_g$ samples uniformly random from the set of valid rankings:

$$\pi_g(a \mid x, q) = \begin{cases} \frac{1}{|\mathcal{A}_g(x,q)|} & \text{if } a \in \mathcal{A}_g(x, q), \\ 0 & \text{otherwise.} \end{cases} \quad (6.27)$$

For the specialization policy space $\Pi_q$, we follow bandit-style online LTR work and take the tabular approach [69]. Documents are scored according to an unbiased estimate of Click-Through-Rate (CTR) on query $q$:

$$\hat{CTR}(d, q) = \frac{1}{|\mathcal{D}_q^{train}|} \sum_{i \in \mathcal{D}_q^{train}} \frac{c_i(d)}{\rho_i}, \quad (6.28)$$

which maximizes the estimated performance (Eq. 6.12). Due to ties there can be multiple valid rankings:

$$\mathcal{A}_q(x, q) = \left\{ a \mid \forall(d_n, d_m) \in x, \left( \hat{CTR}(d_n) > \hat{CTR}(d_m) \to d_n \succ_a d_m \right) \right\}, \quad (6.29)$$

The specialized policy $\pi_q$ also chooses uniformly random from the set of valid rankings:

$$\pi_q(a \mid x, q) = \begin{cases} \frac{1}{|\mathcal{A}_q(x,q)|} & \text{if } a \in \mathcal{A}_q(x,q), \\ 0 & \text{otherwise.} \end{cases} \qquad (6.30)$$

The tabular approach is not restrained by the available features and can produce any possible ranking [138]. Consequently, given enough interactions the tabular approach can perfectly rank items according to relevance. However, CTR cannot be estimated for previously unseen queries and there $\pi_q$ chooses uniformly randomly between all possible rankings. On a query with a single click, $\pi_q$ will place the *once-clicked* item at the front of the ranking. Since clicks are very noisy, this behavior is very risky and hence GENSPEC uses confidence bounds to avoid the deployment of such unsafe behavior.

## 6.5 Experimental Results and Discussion

### 6.5.1 Behavior of the GENSPEC meta-policy

First, we consider the behavior of GENSPEC compared with pure generalization or pure specialization policies. Figures 6.2 and 6.3 show the performance of (i) GENSPEC with different levels of confidence for its bounds ($\epsilon$), along with that of (ii) the logging policy, (iii) the pure generalization policy, and (iv) the pure specialization policies between which the GENSPEC meta-policy chooses. We see that pure generalization requires few clicks to improve over the logging policy but is not able to reach optimal levels of performance. The performance of pure specialization, on the other hand, is initially far below the logging policy. However, after enough clicks have been gathered, performance increases until the optimal ranking is found; when click noise is limited ($\alpha = 0.2$) it reaches perfect performance on all three datasets (Train-NDCG). On the unseen queries where there are no clicks (Test-NDCG), the specialization policy is unable to learn anything and provides random performance (not displayed in Figures 6.2 and 6.3). The initial period of poor performance can be very detrimental to queries that do not receive a large number of clicks. Prior work has found that web-search queries follow a long-tail distribution [113, 115]; White et al. [130] found that 97% of queries received fewer than 10 clicks over six months. For such queries, users may only experience the initial poor performance of pure specialization, and never see the improvements it brings at convergence. This possibility can be a large deterrent from applying pure specialization in practice [131].

Finally, the GENSPEC policy combines properties of both: after a few clicks it deploys the generalization policy and thus outperforms the logging policy; as more clicks are gathered, specialization policies are activated, further improving performance. With $\alpha = 0.2$ the GENSPEC policy with $\epsilon \leq 0.5$ reaches perfect Train-NDCG performance on all three datasets, similar to the pure specialization policy. However, unlike pure specialization the performance of GENSPEC (with $\epsilon > 0$) never drops below the logging policy. Moreover, we never observe the situation where an increase in the number of clicks results in a decrease in mean performance. There is a delay between when the pure specialization policy is the optimal choice and when GENSPEC activates specialization policies. Thus, while the usage of confidence bounds prevents the performance from
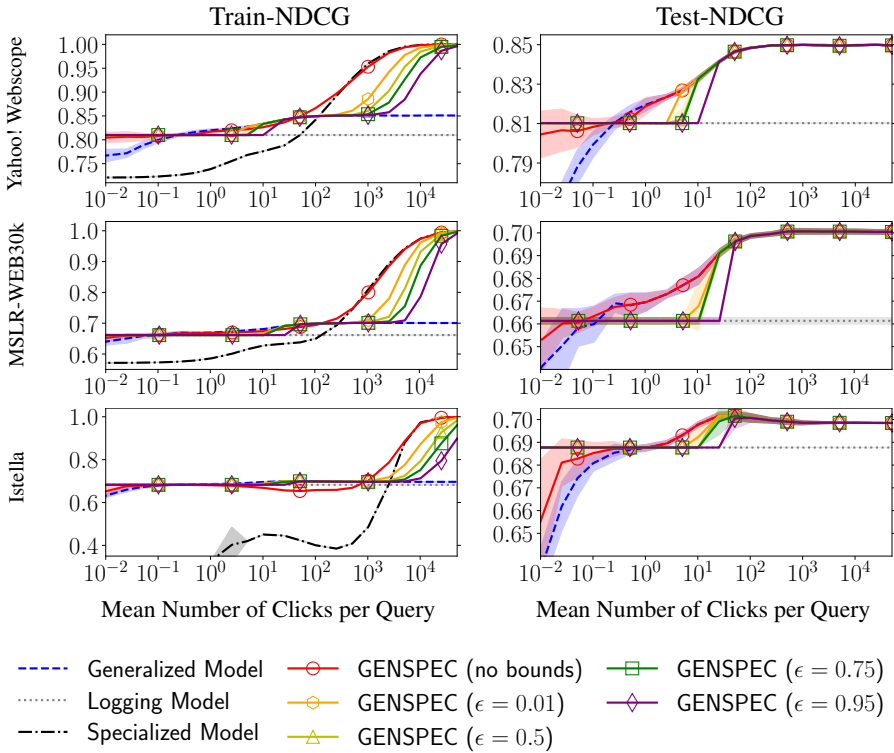
Figure 6.2: Performance of GENSPEC with varying levels of confidence, compared to pure generalization and pure specialization, on clicks generated with $\alpha = 0.2$. We separate queries on the training set (Train-NDCG) that have received clicks, and queries on the test set (Test-NDCG) that do not receive any clicks. Clicks are spread uniformly over the training set, the x-axis indicates the total number of clicks divided by the number of training queries. Results are an average of 10 runs; shaded area indicates the standard deviation.

dropping below the level of the logging policy, it does so at the cost of this delay. When GENSPEC does not use any bounds it deploys specialized policies earlier, however, in some cases these deployments result in worse performance than the logging policy, albeit less than pure specialization. In all our observed results, a confidence of $\epsilon = 0.01$ was enough to prevent any decreases in performance.

To conclude, our experimental results show that the GENSPEC meta-policy combines the high-performance at convergence of specialization and the safe robustness of generalization. In contrast to pure specialization, which results in very poor performance when not enough clicks have been gathered, GENSPEC effectively avoids incorrect deployment and under our tested conditions it never performs worse than the logging policy. Meanwhile, GENSPEC achieves considerable gains in performance at convergence, in contrast with pure generalization. Therefore, we conclude that GENSPEC is
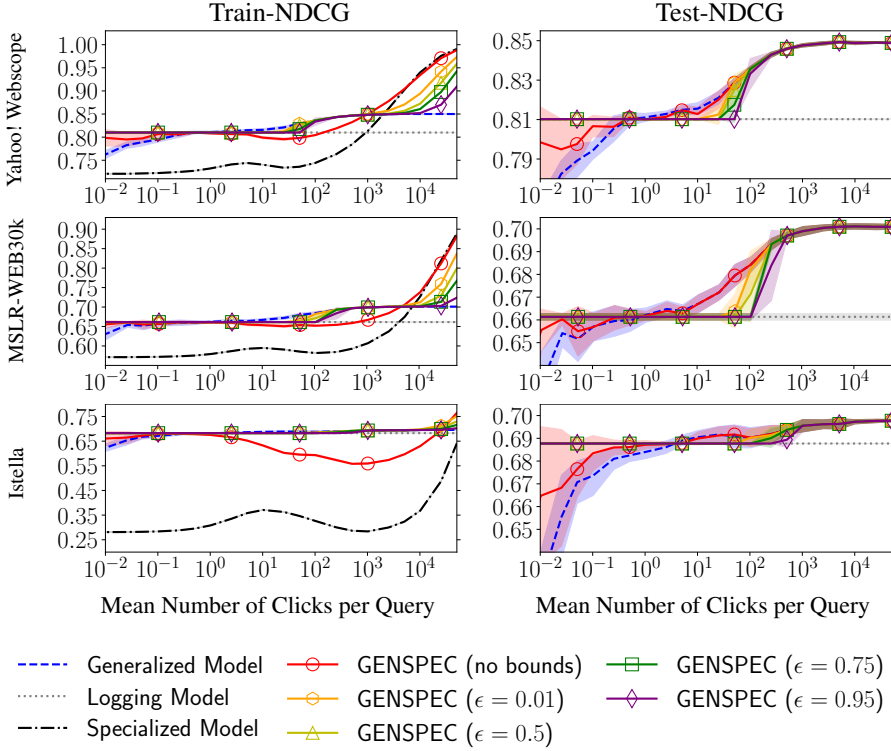
Figure 6.3: Performance of GENSPEC with varying levels of confidence, compared to pure generalization and pure specialization, on clicks generated with $\alpha = 0.025$. Notation is the same as in Figure 6.2.

the best choice in situations where periods of poor performance have to be avoided [131] or when not all queries receive large numbers of clicks [130].

## 6.5.2 Effectiveness of relative bounding

GENSPEC is not the first method that deploys policies based on confidence bounds. As discussed in Section 6.3, Jagerman et al. [51] previously introduced the SEA algorithm. SEA chooses between deploying a generalizing policy or keeping the logging policy in deployment, by bounding both the performance of the logging and generalization policy. When the upper bound of the logging policy is less than the lower bound of the generalizing policy, SEA deploys the latter. The big differences with GENSPEC are that SEA (i) uses two bounds to confidently estimate if one policy outperforms another, and (ii) does not consider specialization policies. Because GENSPEC directly bounds relative performance, its comparisons only use a single bound and thus we expect it to be more efficient w.r.t. the number of clicks required than SEA (see Appendix 6.B for a formal analysis).

For a fair comparison, we adapt SEA to choose between the same policies as
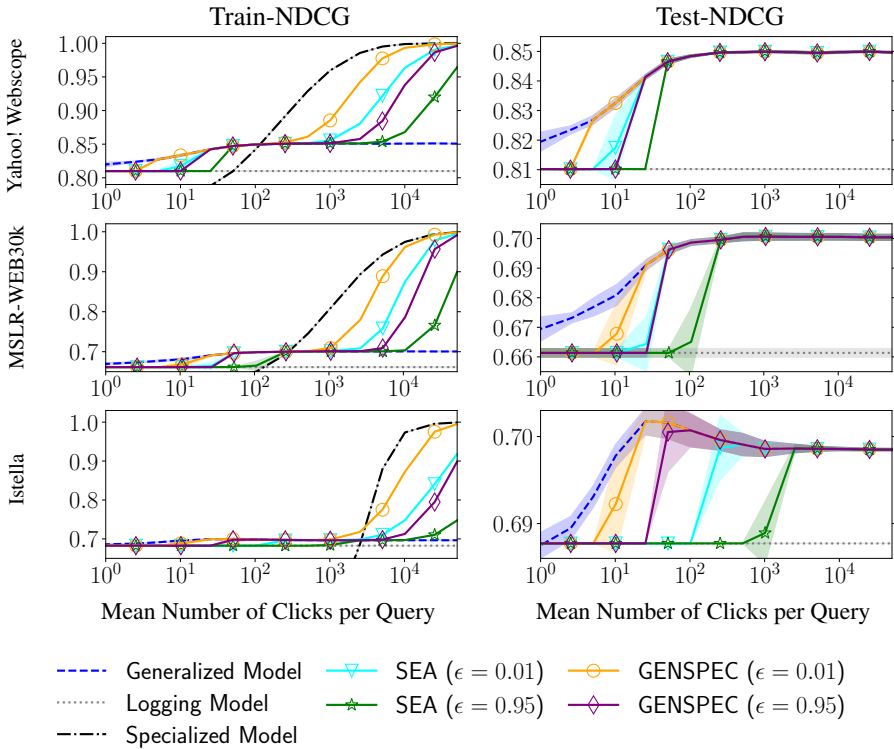
Figure 6.4: GENSPEC compared to a meta-policy using the SEA bounds (see Section 6.5.2), on clicks generated with $\alpha = 0.2$. Notation is the same as in Figure 6.2.

GENSPEC and provide it with the same click data. Figure 6.4 and 6.5 display the results of this comparison. Across all settings, GENSPEC deploys policies much earlier than SEA with the same level of confidence. While they converge at the same levels of performance, GENSPEC requires considerably less data, e.g., on the Istella dataset with $\alpha = 0.025$ GENSPEC deploys with 10 times less data. Thus, we conclude that the relative bounds of GENSPEC are much more efficient than the existing bounding approach of SEA.

## 6.5.3 Comparison with counterfactual LTR

Obvious baselines for our experiments are methods from the counterfactual LTR field [5, 58, 127]. In our setting where the observance probabilities are given, all these methods reduce to Oosterhuis and de Rijke [86]'s method (see Chapter 5), i.e., the method used to optimize the pure generalization policy in Figures 6.2 and 6.3. Thus, the comparison between GENSPEC and pure generalization is effectively a comparison between GENSPEC and state-of-the-art counterfactual LTR. As expected, we see that GENSPEC reaches the same performance on previously unseen queries (Test-NDCG); but on queries with clicks (Train-NDCG) GENSPEC outperforms standard
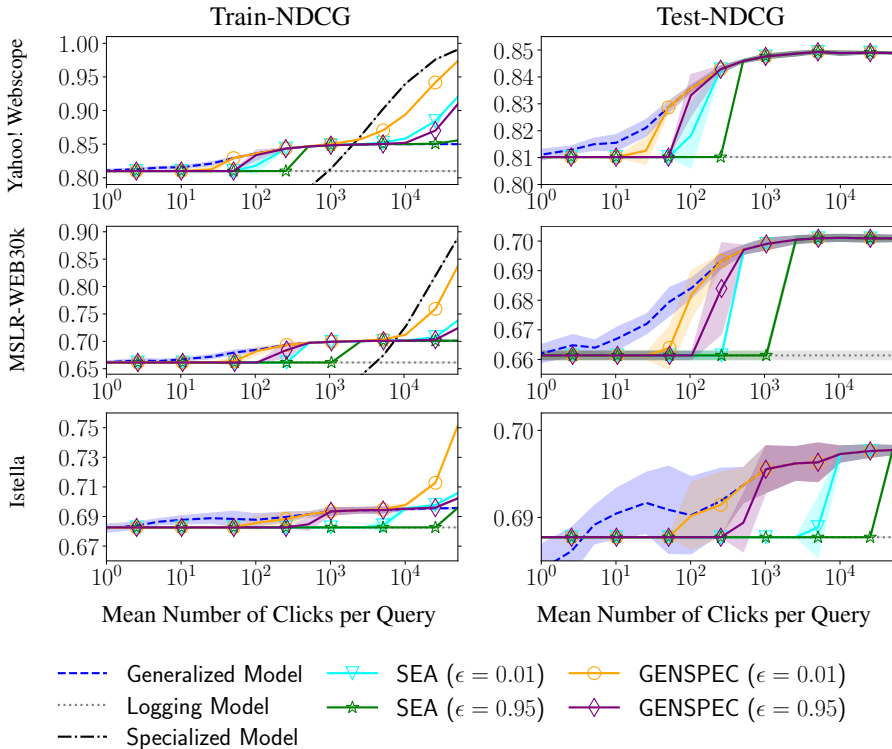
Figure 6.5: GENSPEC compared to a meta-policy using the SEA bounds (see Section 6.5.2), on clicks generated with $\alpha = 0.025$. Notation is the same as in Figure 6.2.

counterfactual LTR by enormous amounts once many clicks have been gathered. Again, there is a small delay between the moment the generalization policy outperforms the logging policy and when GENSPEC deploys it. Since this observed delay is very short, this downside seems to be heavily outweighed by the large increases in performance in Train-NDCG. Thus, we conclude that GENSPEC is preferable over the existing counterfactual LTR approaches, due to its ability to incorporate highly specialized models in its policy.

## 6.5.4 Comparison with online LTR bandits

Other related methods are online LTR bandit algorithms [61, 68]. Unlike counterfactual LTR, these bandit methods learn using online interventions: at each timestep they choose which ranking to display to users. Thus, they have some control over the interactions they receive, and attempt to display rankings that will benefit the learning process the most. As baselines we use the hotfix algorithm [138] and the Position-Based Model algorithm (PBM) [69]. The hotfix algorithm is a very general approach, it completely randomly shuffles the top-$n$ items and ranks them based on pairwise preferences inferred from clicks. The main downside of the hotfix approach is that it
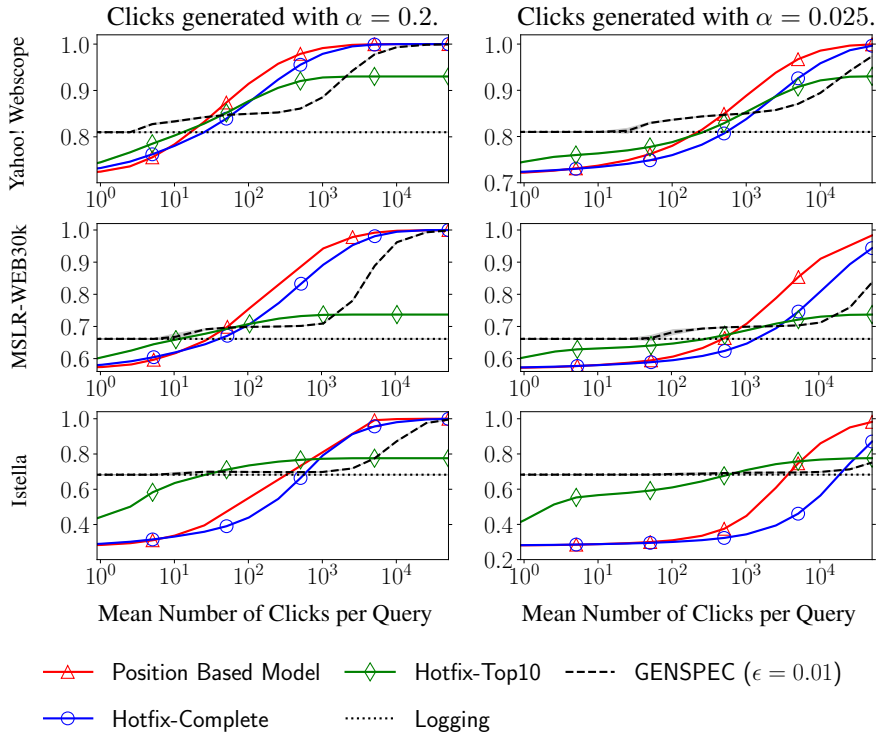
Figure 6.6: GENSPEC compared to various online LTR bandits (see Section 6.5.4). Notation is the same as in Figure 6.2.

can be very detrimental to the user experience due to the randomization.[2] We apply two versions of the hotfix algorithm, one for top-10 reranking to minimize randomization and another for reranking the complete ranking. PBM is perfectly suited for our task as it makes the same assumptions about user behavior as our experimental setting. We apply PMB-PIE [69], which results in PBM always displaying the ranking it expects to perform best, thus attempting to maximize the user experience during learning. These methods are very similar to our specialization policies: the bandit baselines memorize the best rankings and do not depend on features at all. Consequently, their learned policies cannot be applied to previously unseen queries.

Figure 6.6 displays the results for this comparison. We see that when $\alpha = 0.2$ Hotfix-Complete, PBM and GENSPEC all reach perfect Train-NCDG; however, Hotfix-Complete and PBM reach convergence much earlier than GENSPEC. We attribute this difference to three causes: (i) the online interventions of the bandit baselines, (ii) GENSPEC only uses 70% of the available data for training ($\mathcal{D}^{train}$) whereas the bandit baselines use everything, and (iii) the delay in deployment added by GENSPEC's usage of confidence bounds. Similar to the pure specialization policies, the earlier

---

[2]We only report the performance of the ranking produced by the hotfix baseline, not of the randomized rankings used to gather clicks.

moment of convergence of the bandit baselines comes at the cost of an initial period of very poor performance. We conclude that if only the moment of reaching optimal performance matters, PBM is the best choice of method. However, if periods of poor performance should be avoided [131], or if some queries may not receive large numbers of clicks [130], GENSPEC is the better choice. An additional advantage is that GENSPEC is a counterfactual method and does not have to be applied online like the bandit baselines.

### 6.5.5 Feature-based online LTR

Besides the bandit baselines discussed in Section 6.5.4, feature-based methods for online LTR also exist [82, 111, 126, 132]. A direct experimental comparison with these methods is beyond the scope of this chapter. However, previous work has already compared these methods with each other [84] and the state-of-the-art method with counterfactual LTR [50]. Based on the latter work by Jagerman et al. [50] we do not expect considerable differences between these online LTR methods and counterfactual LTR in our settings. Therefore, we expect that a comparison would lead to similar results as discussed in Section 6.5.3.

## 6.6 GENSPEC for Contextual Bandits

So far we have discussed GENSPEC for counterfactual LTR. We will now show that it is also applicable to the broader contextual bandit problem. Instead of a query $q$, we now keep track of an arbitrary context $z \in \{1, 2, \ldots\}$ where

$$x_i, z_i \sim P(x, z). \tag{6.31}$$

Data is gathered using a logging policy $\pi_0$:

$$a_i \sim \pi_0(a \mid x_i, z_i). \tag{6.32}$$

However, unlike the LTR case, the rewards $r_i$ are observed directly:

$$r_i \sim P(r \mid a_i, x_i, z_i). \tag{6.33}$$

With the propensities

$$\rho_i = \pi_0(a_i \mid x_i, z_i) \tag{6.34}$$

the data is:

$$\mathcal{D} = \left\{ (r_i, a_i, \rho_i, x_i, z_i) \right\}_{i=1}^{N}; \tag{6.35}$$

for specialization the data is filtered per context $z$:

$$\mathcal{D}_z = \left\{ (r_i, a_i, \rho_i, x_i, z_i) \in \mathcal{D} \mid z_i = z \right\}. \tag{6.36}$$

Again, data for training $\mathcal{D}^{train}$ and for policy selection $\mathcal{D}^{sel}$ are separated. The reward is estimated with an IPS estimator:

$$\hat{\mathcal{R}}(\pi \mid \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \frac{r_i}{\rho_i} \pi(a_i \mid x_i, z_i). \tag{6.37}$$

With the policy spaces $\Pi_g$ and $\Pi_z$, the policy for generalization is:

$$\pi_g = \arg\max_{\pi \in \Pi_g} \hat{\mathcal{R}}(\pi \mid \mathcal{D}^{train}); \tag{6.38}$$

per context $z$, the specialization policy is:

$$\pi_z = \arg\max_{\pi \in \Pi_z} \hat{\mathcal{R}}(\pi \mid \mathcal{D}_z^{train}). \tag{6.39}$$

The difference between two policies is estimated by:

$$\hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) = \hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}). \tag{6.40}$$

We differ from the LTR approach by estimating the bounds using:

$$R_i = \frac{r_i}{\rho_i} \left( \pi_1(a_i \mid x_i, z_i) - \pi_2(a_i \mid x_i, z_i) \right). \tag{6.41}$$

Following Thomas et al. [119], the confidence bounds are:

$$
\begin{aligned}
CB(\pi_1, \pi_2 \mid \mathcal{D}) = {}& \frac{7b \ln\left(\frac{2}{1-\epsilon}\right)}{3(|\mathcal{D}| - 1)} \\
& + \frac{1}{|\mathcal{D}|} \sqrt{ \frac{2|\mathcal{D}| \ln\left(\frac{2}{1-\epsilon}\right)}{|\mathcal{D}| - 1} \sum_{i \in \mathcal{D}} \left( R_i - \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) \right)^2 },
\end{aligned}
\tag{6.42}
$$

where $b$ is the maximum possible value for $R_i$. This results in the lower bound:

$$LCB(\pi_1, \pi_2 \mid \mathcal{D}) = \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) - CB(\pi_1, \pi_2 \mid \mathcal{D}), \tag{6.43}$$

which is used by the GENSPEC meta-policy:

$$
\pi_{GS}(a \mid x, z) = \\
\begin{cases}
\pi_z(a \mid x, z), & \text{if } \left( LCB(\pi_z, \pi_g | \mathcal{D}_z^{sel}) > 0 \wedge LCB(\pi_z, \pi_0 | \mathcal{D}_z^{sel}) > 0 \right), \\
\pi_g(a \mid x, z), & \text{if } \left( LCB(\pi_z, \pi_g | \mathcal{D}_z^{sel}) \leq 0 \wedge LCB(\pi_g, \pi_0 | \mathcal{D}^{sel}) > 0 \right), \\
\pi_0(a \mid x, z), & \text{otherwise.}
\end{cases}
\tag{6.44}
$$

As such, GENSPEC can be applied to the contextual bandit problem for any arbitrary choice of context $z$.

## 6.7  Conclusion

In this chapter we have introduced the Generalization and Specialization (GENSPEC) framework for contextual bandit problems. For an arbitrary choice of contexts it simultaneously learns a general policy to perform well across all contexts, and many specialized policies each optimized for a single context. Then, per context the GENSPEC meta-policy uses high-confidence bounds to choose between deploying the logging policy,

the general policy, or a specialized policy. As a result, GENSPEC combines the robust safety of a general policy with the high-performance of a successfully specialized policy.

We have shown how GENSPEC can be applied to query-specialization for counterfactual LTR. Our results show that GENSPEC combines the high performance of specialized policies on queries with sufficiently many interactions, with the robust performance on queries that were previously unseen or where little data is available. Thus, it avoids the low performance at convergence of feature-based models underlying the general policy, and the initial poor performance of the tabular models underlying the specialized policies. We expect that GENSPEC can be used for other types of specialization by choosing different context divisions, i.e., personalization for LTR is a promising choice.

With these findings we can answer thesis research question **RQ6** positively: Using GENSPEC we can combine the specialization ability of bandit-style online LTR with the robust generalization of feature-based LTR. As a result, the choice between specialization and generalization can now be made in a principled, theoretically-grounded manner. For the LTR field this means that bandit-style LTR and feature-based LTR can now be seen as complementary, instead of a mutually exclusive choice.

Future work could explore other contextual bandit problems and choices for context. Additionally, we hope that the robust safety of GENSPEC further incites the application of bandit algorithms in practice.

While this chapter considered GENSPEC for counterfactual LTR, Chapter 8 introduces a novel method that is both effective at counterfactual LTR and online LTR. With only small adaptations the contributions of both chapters could be combined, thus potentially resulting in GENSPEC for both online and counterfactual LTR. Future work could investigate the effectiveness of this possible combined approach.

## 6.A Proof of Unbiasedness for Counterfactual Learning to Rank

This section will prove that the IPS estimate $\hat{\mathcal{R}}$ (Eq. 6.11) can be used to unbiasedly optimize the true reward $\mathcal{R}$ (Eq. 6.1), as claimed in Section 6.2.2. For this proof we rely on the following assumptions: (i) LTR metrics are linear combinations of item relevances (Eq. 6.2), (ii) the assumption that clicks never occur on unobserved items (Eq. 6.7), and (iii) click probabilities (conditioned on observance) are proportional to relevance (Eq. 6.8).

First, we consider the expected value for an observed click $c_i(d)$ using Eq. 6.7; for brevity we write $r(d) = r(d \mid x_i, q_i)$:

$$
\begin{aligned}
\mathbb{E}_{o_i, a_i} & \big[ c_i(d) \big] \\
&= \mathbb{E}_{a_i} \Big[ P\big( c_i(d) = 1 | o_i(d) = 1, r(d) \big) \cdot P\big( o_i(d) = 1 | a_i \big) \Big] \\
&= P\big( c_i(d) = 1 \mid o_i(d) = 1, r(d) \big) \cdot \Bigg( \sum_{a \in \pi_0} P\big( o_i(d) = 1 \mid a \big) \cdot \pi_0(a \mid x_i, q_i) \Bigg) \\
&= \rho_i(d) \cdot P\big( c_i(d) = 1 \mid o_i(d) = 1, r(d) \big).
\end{aligned}
$$
(6.45)

Then, consider the expected value for the IPS estimator, and note that $a_i$ is a historically observed action and that $a$ is the action being evaluated:

$$
\begin{aligned}
\mathbb{E}_{o_i, a_i} \big[ \hat{\Delta}(a \mid c_i, \rho_i) \big] &= \mathbb{E}_{o_i, a_i} \Bigg[ \sum_{d \in a} \frac{\lambda\big( rank(d \mid a) \big) \cdot c_i(d)}{\rho_i(d)} \Bigg] \\
&= \sum_{d \in a} \frac{\rho_i(d)}{\rho_i(d)} \cdot \lambda\big( rank(d \mid a) \big) \cdot P\big( c_i(d) = 1 \mid o_i(d) = 1, r(d) \big) \\
&= \sum_{d \in a} \lambda\big( rank(d \mid a) \big) \cdot P\big( c_i(d) = 1 \mid o_i(d) = 1, r(d) \big).
\end{aligned}
$$
(6.46)

This step assumes that $\rho_i(d) > 0$, i.e., that every item has a non-zero probability of being examined [58]. While $\mathbb{E}_{o_i, a_i}[\hat{\Delta}(a \mid c_i, \rho_i)]$ and $\Delta(a \mid x_i, q_i, r)$ are not necessarily equal, using Eq. 6.8 we see that they are proportional with some offset $C$:

$$
\begin{aligned}
\mathbb{E}_{o_i, a_i} \big[ \hat{\Delta}(a \mid c_i, \rho_i) \big] &\propto \Bigg( \sum_{d \in a} \lambda\big( rank(d \mid a) \big) \cdot r(d) \Bigg) + C \\
&= \Delta(a \mid x_i, q_i, r) + C,
\end{aligned}
$$
(6.47)

where $C$ is a constant: $C = \big( \sum_{i=1}^{K} \lambda(i) \big) \cdot \mu$. Therefore, in expectation $\hat{\mathcal{R}}$ and $\mathcal{R}$ are also proportional with the same constant offset:

$$
\mathbb{E}_{o_i, a_i} \big[ \hat{\mathcal{R}}(\pi \mid \mathcal{D}) \big] \propto \mathcal{R}(\pi) + C.
$$
(6.48)

Consequently, the estimator can be used to unbiasedly estimate the preference between two policies:

$$\mathbb{E}_{o_i,a_i}\big[\hat{\mathcal{R}}(\pi_1 \mid \mathcal{D})\big] < \mathbb{E}_{o_i,a_i}\big[\hat{\mathcal{R}}(\pi_2 \mid \mathcal{D})\big] \Leftrightarrow \mathcal{R}(\pi_1) < \mathcal{R}(\pi_2). \qquad (6.49)$$

Moreover, this implies that maximizing the estimated performance unbiasedly optimizes the actual reward:

$$\arg\max_{\pi} \mathbb{E}_{o_i,a_i}\big[\hat{\mathcal{R}}(\pi \mid \mathcal{D})\big] = \arg\max_{\pi} \mathcal{R}(\pi). \qquad (6.50)$$

This concludes our proof. We have shown that $\hat{\mathcal{R}}$ is suitable for counterfactual evaluation since it can unbiasedly identify if a policy outperforms another (Eq. 6.49) and, furthermore, that $\hat{\mathcal{R}}$ can be used for unbiased LTR, i.e., it can be used to find the optimal policy (Eq. 6.50).

## 6.B  Efficiency of Relative Bounding by GENSPEC

Our experimental results showed that GENSPEC chooses between policies more efficiently than when using SEA bounds [51]. In other words, when one policy has higher performance than another, the relative bounds of GENSPEC require less data to be certain about this difference than the SEA bounds. In this section, we will prove that the relative bounds of GENSPEC are more efficient than the SEA bounds when the covariance between the reward estimates of two policies is positive:

$$\mathrm{cov}\big(\hat{\mathcal{R}}(\pi_1|\mathcal{D}), \hat{\mathcal{R}}(\pi_2|\mathcal{D})\big) > 0. \qquad (6.51)$$

This means that GENSPEC will deploy a policy earlier than SEA if there is high covariance, since both estimates are based on the same interaction data $\mathcal{D}$ a high covariance is very likely.

Let us first consider when GENSPEC deploys a policy: Deployment by GENSPEC depends on whether a relative confidence bound is greater than the estimated difference in performance (cf. Eq. 6.24). For two policies $\pi_1$ and $\pi_2$ deployment happens when:

$$\hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}) - CB(\pi_1, \pi_2 \mid \mathcal{D}) > 0. \qquad (6.52)$$

Thus the bound has to be smaller than the estimated performance difference:

$$CB(\pi_1, \pi_2 \mid \mathcal{D}) < \hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}). \qquad (6.53)$$

In contrast, SEA does not use a single bound, but bounds the performance of both policies. For clarity, we reformulate the SEA bound in our notation. First we have $R_{i,d}^{\pi_j}$ the observed reward for an item $d$ at interaction $i$ for policy $\pi_j$:

$$R_{i,d}^{\pi_j} = \frac{c_i(d)}{\rho_i(d)} \sum_{a \in \pi_j} \pi_j(a|x_i, q_i) \cdot \lambda\big(rank(d|a)\big). \qquad (6.54)$$

Then we have a $\nu^{\pi_j}$ for each policy:

$$\nu^{\pi_j} = \frac{2|\mathcal{D}|K \ln\left(\frac{2}{1-\epsilon}\right)}{|\mathcal{D}|K - 1} \sum_{(i,d)\in\mathcal{D}} \left(K \cdot R_{i,d}^{\pi_j} - \hat{\mathcal{R}}(\pi_j \mid \mathcal{D})\right)^2,$$

which we use to note the confidence bound for a single policy $\pi_j$:

$$CB(\pi_j \mid \mathcal{D}) = \frac{7Kb \ln\left(\frac{2}{1-\epsilon}\right)}{3(|\mathcal{D}|K - 1)} + \frac{1}{|\mathcal{D}|K} \cdot \sqrt{\nu^{\pi_j}}. \tag{6.55}$$

We note that the $b$ parameter has the same value for both the relative and single confidence bounds. SEA chooses between policies by comparing their upper and lower confidence bounds:

$$\hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - CB(\pi_1 \mid \mathcal{D}) > \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}) + CB(\pi_2 \mid \mathcal{D}). \tag{6.56}$$

In this case, the summation of the bounds has to be smaller than the estimated performance difference:

$$CB(\pi_1 \mid \mathcal{D}) + CB(\pi_2 \mid \mathcal{D}) < \hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}). \tag{6.57}$$

We can now formally describe under which condition GENSPEC is more efficient than SEA: by combining Eq. 6.53 and Eq. 6.57, we see that relative bounding is more efficient when:

$$CB(\pi_1, \pi_2 \mid \mathcal{D}) < CB(\pi_1 \mid \mathcal{D}) + CB(\pi_2 \mid \mathcal{D}). \tag{6.58}$$

We notice that $\mathcal{D}$, $K$, $b$ and $\epsilon$ have the same value for both confidence bounds, thus we only require:

$$\sqrt{\nu} < \sqrt{\nu^{\pi_1}} + \sqrt{\nu^{\pi_2}}. \tag{6.59}$$

If we assume that $\mathcal{D}$ is sufficiently large, we see that $\sqrt{\nu}$ approximates the standard deviation scaled by some constant:

$$\sqrt{\nu} \approx C \cdot \sqrt{\mathrm{var}\left(\hat{\delta}(\pi_1, \pi_2 | \mathcal{D})\right)}, \tag{6.60}$$

where the constant is: $C = \sqrt{\frac{2|\mathcal{D}|^2 K^2 \ln\left(\frac{2}{1-\epsilon}\right)}{|\mathcal{D}|K-1}}$. Since the purpose of the bounds is to prevent deployment until enough certainty has been gained, we think it is safe to assume that $\mathcal{D}$ is large enough for this approximation before any deployment takes place.

To keep our notation concise, we use the following: $\hat{\delta} = \hat{\delta}(\pi_1, \pi_2 | \mathcal{D})$, $\hat{\mathcal{R}}_1 = \hat{\mathcal{R}}(\pi_1 | \mathcal{D})$, and $\hat{\mathcal{R}}_2 = \hat{\mathcal{R}}(\pi_2 | \mathcal{D})$. Using the same approximations for $\sqrt{\nu^{\pi_1}}$ and $\sqrt{\nu^{\pi_2}}$ we get:

$$\sqrt{\mathrm{var}(\hat{\delta})} < \sqrt{\mathrm{var}(\hat{\mathcal{R}}_1)} + \sqrt{\mathrm{var}(\hat{\mathcal{R}}_2)}. \tag{6.61}$$

By making use of the Cauchy-Schwarz inequality, we can derive the following lower bound:

$$\sqrt{\mathrm{var}(\hat{\mathcal{R}}_1) + \mathrm{var}(\hat{\mathcal{R}}_2)} \leq \sqrt{\mathrm{var}(\hat{\mathcal{R}}_1)} + \sqrt{\mathrm{var}(\hat{\mathcal{R}}_2)}. \tag{6.62}$$

Therefore, the relative bounding of GENSPEC must be more efficient when the following is true:

$$\text{var}(\hat{\delta}) < \text{var}(\hat{\mathcal{R}}_1) + \text{var}(\hat{\mathcal{R}}_2), \tag{6.63}$$

i.e. the variance of the relative estimator must be less than the sum of the variances of the estimators for the individual policies. Finally, by rewriting $\text{var}(\hat{\delta})$ to:

$$\text{var}(\hat{\delta}) = \text{var}(\hat{\mathcal{R}}_1 - \hat{\mathcal{R}}_2) = \text{var}(\hat{\mathcal{R}}_1) + \text{var}(\hat{\mathcal{R}}_2) - 2\text{cov}(\hat{\mathcal{R}}_1, \hat{\mathcal{R}}_2), \tag{6.64}$$

we see that the relative bounds of GENSPEC are more efficient than the multiple bounds of SEA if the covariance between $\hat{\mathcal{R}}_1$ and $\hat{\mathcal{R}}_2$ is positive:

$$\text{cov}(\hat{\mathcal{R}}_1, \hat{\mathcal{R}}_2) > 0. \tag{6.65}$$

Remember that both estimates are based on the same interaction data: $\hat{\mathcal{R}}_1 = \hat{\mathcal{R}}(\pi_1|\mathcal{D})$, and $\hat{\mathcal{R}}_2 = \hat{\mathcal{R}}(\pi_2|\mathcal{D})$. Therefore, they are based on the same clicks and propensities scores, thus it is extremely likely that the covariance between the estimates is positive. Correspondingly, it is also extremely likely that the relative bounds of GENSPEC are more efficient than the bounds used by SEA.

## 6.C   Notation Reference for Chapter 6

| Notation | Description |
|----------|-------------|
| $K$ | the number of items that can be displayed in a single ranking |
| $i$ | an iteration number |
| $q$ | a user-issued query |
| $x$ | contextual information, i.e., additional features |
| $d$ | an item to be ranked |
| $a$ | a ranked list |
| $\pi$ | a ranking policy |
| $\pi(a \mid q)$ | the probability that policy $\pi$ displays ranking $a$ for query $q$ |
| $r(d \mid x, q)$ | the relevance of item $d$ w.r.t. query $q$ given context $x$ |
| $\lambda\big(rank(d \mid a)\big)$ | a metric function that weights items depending on their rank |
| $\mathcal{D}$ | the available interaction data |
| $c_i(d)$ | a function indicating item $d$ was clicked at iteration $i$ |
| $o_i(d)$ | a function indicating item $d$ was observed at iteration $i$ |

# 7

# Taking the Counterfactual Online: Efficient and Unbiased Online Evaluation for Ranking

Counterfactual evaluation can estimate Click-Through-Rate (CTR) differences between ranking systems based on historical interaction data, while mitigating the effect of position bias and item-selection bias. In contrast, online evaluation methods, designed for ranking, estimate performance differences between ranking systems by showing interleaved rankings to users and observing their clicks. We are curious to find out whether the online interventions of online evaluation methods truly result in more efficient evaluation, and additionally, whether the popular interleaving methods are truly unbiased w.r.t. biases such as position bias. Accordingly this chapter will consider the following two thesis research questions:

**RQ7** Can counterfactual evaluation methods for ranking be extended to perform efficient and effective online evaluation?

**RQ8** Are existing interleaving methods truly capable of unbiased evaluation w.r.t. position bias?

We introduce the novel Logging-Policy Optimization Algorithm (LogOpt), which optimizes the policy for logging data so that the counterfactual estimate has minimal variance. As minimizing variance leads to faster convergence, LogOpt increases the data-efficiency of counterfactual estimation. LogOpt turns the counterfactual approach – which is indifferent to the logging policy – into an online approach, where the algorithm decides what rankings to display. We prove that, as an online evaluation method, LogOpt is unbiased w.r.t. position and item-selection bias, unlike existing interleaving methods. Furthermore, we perform large-scale experiments by simulating comparisons between *thousands* of rankers. Our results show that while interleaving methods make systematic errors, LogOpt is as efficient as interleaving without being biased. Lastly, we provide a formal proof that shows interleaving methods are not unbiased w.r.t. position bias.

---

This chapter was published as [85]. Appendix 7.C gives a reference for the notation used in this chapter.

# 7.1   Introduction

Evaluation is essential for the development of search and recommendation systems [45, 64]. Before any ranking model is widely deployed it is important to first verify whether it is a true improvement over the currently-deployed model. A traditional way of evaluating relative differences between systems is through A/B testing, where part of the user population is exposed to the current system ("control") and the rest to the altered system ("treatment") during the same time period. Differences in behavior between these groups can then indicate if the alterations brought improvements, e.g., if the treatment group showed a higher CTR or more revenue was made with this system [18].

Interleaving has been introduced in Information Retrieval (IR) as a more efficient alternative to A/B testing [56]. Interleaving algorithms take the rankings produced by two ranking systems, and for each query create an interleaved ranking by combining the rankings from both systems. Clicks on the interleaved rankings directly indicate relative differences. Repeating this process over a large number of queries and averaging the results, leads to an estimate of which ranker would receive the highest CTR [44]. Previous studies have found that interleaving requires fewer interactions than A/B testing, which enables them to make consistent comparisons in a much shorter timespan [18, 110].

More recently, counterfactual evaluation for rankings has been proposed by Joachims et al. [58] to evaluate a ranking model based on clicks gathered using a different model. By correcting for the position bias introduced during logging, the counterfactual approach can unbiasedly estimate the CTR of a new model on historical data. To achieve this, counterfactual evaluation makes use of Inverse Propensity Scoring (IPS), where clicks are weighted inversely to the probability that a user examined them during logging [127]. A big advantage compared to interleaving and A/B testing, is that counterfactual evaluation does not require online interventions.

In this chapter, we show that no existing interleaving method is truly unbiased: they are not guaranteed to correctly predict which ranker has the highest CTR. On two different industry datasets, we simulate a total of 1,000 comparisons between 2,000 different rankers. In our setup, interleaving methods converge on the wrong answer for at least 2.2% of the comparisons on both datasets. A further analysis shows that existing interleaving methods are unable to reliably estimate CTR differences of around 1% or lower. Therefore, in practice these systematic errors are expected to impact situations where rankers with a very similar CTR are compared.

We propose a novel online evaluation algorithm: Logging-Policy Optimization Algorithm (LogOpt). LogOpt extends the existing unbiased counterfactual approach, and turns it into an online approach. LogOpt estimates which rankings should be shown to the user, so that the variance of its CTR estimate is minimized. In other words, it attempts to learn the logging-policy that leads to the fastest possible convergence of the counterfactual estimation. Our experimental results indicate that our novel approach is as efficient as any interleaving method or A/B testing, without having a systematic error. As predicted by the theory, we see that the estimates of our approach converge on the true CTR difference between rankers. Therefore, we have introduced the first online evaluation method that combines high efficiency with unbiased estimation.

The main contributions of this chapter are:

1. The first logging-policy optimization method for minimizing the variance in counter-factual CTR estimation.

2. The first unbiased online evaluation method that is as efficient as state-of-the-art interleaving methods.

3. A large-scale analysis of existing online evaluation methods that reveals a previously unreported bias in interleaving methods.

## 7.2 Preliminaries: Ranker Comparisons

The overarching goal of ranker evaluation is to find the ranking model that provides the best rankings. For the purposes of this chapter, we will define the quality of a ranker in terms of the number of clicks it is expected to receive. Let $R$ indicate a ranking and let $\mathbb{E}[\text{CTR}(R)] \in \mathbb{R}_{\geq 0}$ be the expected number of clicks a ranking receives after being displayed to a user. We consider ranking $R_1$ to be better than $R_2$ if in expectation it receives more clicks: $\mathbb{E}[\text{CTR}(R_1)] > \mathbb{E}[\text{CTR}(R_2)]$. We will represent a ranking model by a policy $\pi$, with $\pi(R \mid q)$ as the probability that $\pi$ displays $R$ for a query $q$. With $P(q)$ as the probability of a query $q$ being issued, the expected number of clicks received under a ranking model $\pi$ is:

$$\mathbb{E}[\text{CTR}(\pi)] = \sum_q P(q) \sum_R \mathbb{E}[\text{CTR}(R)]\pi(R \mid q). \tag{7.1}$$

Our goal is to discover the $\mathbb{E}[\text{CTR}]$ difference between two policies:

$$\Delta(\pi_1, \pi_2) = \mathbb{E}[\text{CTR}(\pi_1)] - \mathbb{E}[\text{CTR}(\pi_2)]. \tag{7.2}$$

We recognize that to correctly identify if one policy is better than another, we merely need a corresponding binary indicator:

$$\Delta_{bin}(\pi_1, \pi_2) = \text{sign}\big(\Delta(\pi_1, \pi_2)\big). \tag{7.3}$$

However, in practice the magnitude of the differences can be very important, for instance, if one policy is computationally much more expensive while only having a slightly higher $\mathbb{E}[\text{CTR}]$, it may be preferable to use the other in production. Therefore, estimating the absolute $\mathbb{E}[\text{CTR}]$ difference is more desirable in practice.

### 7.2.1 User behavior assumptions

Any proof regarding estimators using user interactions must rely on assumptions about user behavior. In this chapter, we assume that only two forms of interaction bias are at play: position bias and item-selection bias.

Users generally do not examine all items that are displayed in a ranking but only click on examined items [20]. As a result, a lower probability of examination for an item also makes it less likely to be clicked. Position bias assumes that only the rank determines the probability of examination [25]. Furthermore, we will assume that given

an examination only the relevance of an item determines the click probability. Let $c(d) \in \{0, 1\}$ indicate a click on item $d$ and $o(d) \in \{0, 1\}$ examination by the user. Then these assumptions result in the following assumed click probability:

$$
\begin{aligned}
P(c(d) = 1 \mid R, q) &= P(o(d) = 1 \mid R) P(c(d) = 1 \mid o(d) = 1, q) \\
&= \theta_{\mathrm{rank}(d \mid R)} \zeta_{d,q}.
\end{aligned}
\tag{7.4}
$$

Here $\mathrm{rank}(d \mid R)$ indicates the rank of $d$ in $R$; for brevity we use $\theta_{\mathrm{rank}(d \mid R)}$ to denote the examination probability:

$$
\theta_{\mathrm{rank}(d \mid R)} = P(o(d) = 1 \mid R),
\tag{7.5}
$$

and $\zeta_{d,q}$ for the conditional click probability:

$$
\zeta_{d,q} = P(c(d) = 1 \mid o(d) = 1, q).
\tag{7.6}
$$

We also assume that item-selection bias is present; this type of bias is an extreme form of position bias that results in zero examination probabilities for some items [86, 92]. This bias is unavoidable in top-$k$ ranking settings, where only the $k \in \mathbb{N}_{>0}$ highest ranked items are displayed. Consequently, any item beyond rank $k$ cannot be observed or examined by the user: $\forall r \in \mathbb{N}_{>0} \, (r > k \rightarrow \theta_r = 0)$. The distinction between item-selection bias and position bias is important because the original counterfactual evaluation method [58] is only able to correct for position bias when no item-selection bias is present [86, 92].

Based on these assumptions, we can now formulate the expected CTR of a ranking:

$$
\mathbb{E}[\mathrm{CTR}(R)] = \sum_{d \in R} P(c(d) = 1 \mid R, q) = \sum_{d \in R} \theta_{\mathrm{rank}(d \mid R)} \zeta_{d,q}.
\tag{7.7}
$$

While we assume this model of user behavior, its parameters are still assumed unknown. Therefore, the methods in this chapter will have to estimate $\mathbb{E}[\mathrm{CTR}]$ without prior knowledge of $\theta$ or $\zeta$.

## 7.2.2 Goal: CTR-estimator properties

Recall that our goal is to estimate the CTR difference between rankers (Eq. 7.2); online evaluation methods do this based on user interactions. Let $\mathcal{I}$ be the set of available user interactions, it contains $N$ tuples of a single (issued) query $q_i$, the corresponding displayed ranking $R_i$, and the observed user clicks $c_i$:

$$
\mathcal{I} = \{(q_i, R_i, c_i)\}_{i=1}^{N}.
\tag{7.8}
$$

Each evaluation method has a different effect on what rankings will be displayed to users. Furthermore, each evaluation method converts each interaction into a single estimate using some function $f$:

$$
x_i = f(q_i, R_i, c_i).
\tag{7.9}
$$

The final estimate is simply the mean over these estimates:

$$\hat{\Delta}(\mathcal{I}) = \frac{1}{N} \sum_{i=1}^{N} x_i = \frac{1}{N} \sum_{i=1}^{N} f(q_i, R_i, c_i). \qquad (7.10)$$

This description fits all existing online and counterfactual evaluation methods for rankings. Every evaluation method uses a different function $f$ to convert interactions into estimates; moreover, online evaluation methods also decide which rankings $R$ to display when collecting $\mathcal{I}$. These two choices result in different estimators. Before we discuss the individual methods, we briefly introduce the three properties we desire of each estimator: consistency, unbiasedness and variance.

- **Consistency** – an estimator is *consistent* if it converges as the number of issued queries $N$ increases. All existing evaluation methods are consistent as their final estimates are means of bounded values.

- **Unbiasedness** – an estimator is *unbiased* if its estimate is equal to the true CTR difference in expectation:

$$\text{Unbiased}(\hat{\Delta}) \Leftrightarrow \mathbb{E}\big[\hat{\Delta}(\mathcal{I})\big] = \Delta(\pi_1, \pi_2). \qquad (7.11)$$

If an estimator is both consistent and unbiased it is guaranteed to converge on the true $\mathbb{E}[\text{CTR}]$ difference.

- **Variance** – the *variance* of an estimator is the expected squared deviation between a single estimate $x$ and the mean $\hat{\Delta}(\mathcal{I})$:

$$\text{Var}(\hat{\Delta}) = \mathbb{E}\Big[\big(x - \mathbb{E}[\hat{\Delta}(\mathcal{I})]\big)^2\Big]. \qquad (7.12)$$

Variance affects the rate of convergence of an estimator; for fast convergence it should be as low as possible.

In summary, our goal is to find an estimator, for the CTR difference between two ranking models, that is consistent, unbiased and has minimal variance.

## 7.3 Existing Online and Counterfactual Evaluation Methods

We describe three families of online and counterfactual evaluation methods for ranking.

### 7.3.1 A/B testing

A/B testing is a well established form of online evaluation to compare a system A with a system B [64]. Users are randomly split into two groups and during the same time period each group is exposed to only one of the systems. In expectation, the only factor that differs between the groups is the exposure to the different systems. Therefore, by

comparing the behavior of each user group, the relative effect each system has can be evaluated.

We will briefly show that A/B testing is unbiased for $\mathbb{E}[\text{CTR}]$ difference estimation. For each interaction either $\pi_1$ or $\pi_2$ determines the ranking, let $A_i \in \{1, 2\}$ indicate the assignment and $A_i \sim P(A)$. Thus, if $A_i = 1$, then $R_i \sim \pi_1(R \mid q)$ and if $A_i = 2$, then $R_i \sim \pi_2(R \mid q)$. Each interaction $i$ is converted into a single estimate $x_i$ by $f_{\text{A/B}}$:

$$x_i = f_{\text{A/B}}(q_i, R_i, c_i) = \left( \frac{\mathbb{1}[A_i = 1]}{P(A = 1)} - \frac{\mathbb{1}[A_i = 2]}{P(A = 2)} \right) \sum_{d \in R_i} c_i(d). \qquad (7.13)$$

We can prove that A/B testing is unbiased, since in expectation each individual estimate is equal to the CTR difference:

$$\mathbb{E}[f_{\text{A/B}}(q_i, R_i, c_i)] = \sum_q P(q) \left( \frac{P(A = 1) \sum_R \pi_1(R \mid q) \mathbb{E}[\text{CTR}(R)]}{P(A = 1)} \right.$$
$$\left. - \frac{P(A = 2) \sum_R \pi_2(R \mid q) \mathbb{E}[\text{CTR}(R)]}{P(A = 2)} \right)$$
$$= \sum_q P(q) \sum_R \mathbb{E}[\text{CTR}(R)] \big( \pi_1(R \mid q) - \pi_2(R \mid q) \big)$$
$$= \mathbb{E}[\text{CTR}(\pi_1)] - \mathbb{E}[\text{CTR}(\pi_2)] = \Delta(\pi_1, \pi_2). \qquad (7.14)$$

Variance is harder to evaluate without knowledge of $\pi_1$ and $\pi_2$. Unless $\Delta(\pi_1, \pi_2) = 0$, some variance is unavoidable since A/B testing alternates between estimating $\text{CTR}(\pi_1)$ and $\text{CTR}(\pi_2)$.

## 7.3.2   Interleaving

Interleaving methods were introduced specifically for evaluation in ranking, as a more efficient alternative to A/B testing [56]. After a query is issued, interleaving methods take the rankings of two competing ranking systems and combine them into a single interleaved ranking. Any clicks on the interleaved ranking can be interpreted as a preference signal between either ranking system. Thus, unlike A/B testing, interleaving does not estimate the CTR of individual systems but a relative preference; the idea is that this allows it to be more efficient than A/B testing.

Each interleaving method attempts to use randomization to counter position bias, without deviating too much from the original rankings so as to maintain the user experience [56]. *Team-draft interleaving* (TDI) randomly selects one ranker to place their top document first, then the other ranker places their top (unplaced) document next [99]. Then it randomly decides the next two documents, and this process is repeated until all documents are placed in the interleaved ranking. Clicks on the documents are attributed to the ranker that placed them. The ranker with the most attributed clicks is inferred to be preferred by the user. *Probabilistic interleaving* (PI) treats each ranking as a probability distribution over documents; at each rank a distribution is randomly selected and a document is drawn from it [41]. After clicks have been received, probabilistic interleaving computes the expected number of clicked

documents per ranking system to infer preferences. *Optimized interleaving* (OI) casts the randomization as an optimization problem, and displays rankings so that if all documents are equally relevant no preferences are found [96].

While every interleaving method attempts to deal with position bias, none is unbiased according to our definition (Section 7.2.2). This may be confusing because previous work on interleaving makes claims of unbiasedness [41, 44, 96]. However, they use different definitions of the term. More precisely, TDI, PI, and OI provably converge on the correct outcome if all documents are equally relevant [41, 44, 96, 99]. Moreover, if one assumes binary relevance and $\pi_1$ ranks all relevant documents equal to or higher than $\pi_2$, the binary outcome of PI and OI is proven to be correct in expectation [44, 96]. However, beyond the confines of these unambiguous cases, we can prove that these methods do not meet our definition of unbiasedness: for every method one can construct an example where it converges on the incorrect outcome. The rankers $\pi_1$, $\pi_2$ and position bias parameters $\theta$ can be chosen so that in expectation the wrong (binary) outcome is estimated; see Appendix 7.A for a proof for each of the three interleaving methods. Thus, while more efficient than A/B testing, interleaving methods make systematic errors in certain circumstances and thus should not be considered to be unbiased w.r.t. CTR differences.

We note that the magnitude of the bias should also be considered. If the systematic error of an interleaving method is minuscule while the efficiency gains are very high, it may still be very useful in practice. Our experimental results (Section 7.6.2) reveal that the systematic error of all three interleaving methods considered becomes very high when comparing systems with a CTR difference of 1% or smaller.

### 7.3.3 Counterfactual evaluation

Counterfactual evaluation is based on the idea that if certain biases can be estimated well, they can also be adjusted [57, 127]. While estimating relevance is considered the core difficulty of ranking evaluation, estimating the position bias terms $\theta$ is very doable. By randomizing rankings, e.g., by swapping pairs of documents [57] or exploiting data logged during A/B testing [4], differences in CTR for the same item on different positions can be observed directly. Alternatively, using Expectation Maximization (EM) optimization [128] or a dual learning objective [5], position bias can be estimated from logged data as well. Once the bias terms $\theta$ have been estimated, logged clicks can be weighted so as to correct for the position bias during logging. Hence, counterfactual evaluation can work with historically logged data. Existing counterfactual evaluation algorithms do not dictate which rankings should be displayed during logging: they do not perform interventions and thus we do not consider them to be online methods.

Counterfactual evaluation assumes that the position bias $\theta$ and the logging policy $\pi_0$ are known, in order to correct for both position bias and item-selection bias. Clicks are gathered with $\pi_0$ which decides which rankings are displayed to the user. We follow Oosterhuis and de Rijke [86] (see Chapter 5) and use as propensity scores the probability of observance in expectation over the displayed rankings:

$$\rho(d \mid q) = \mathbb{E}_R\big[P(o(d) = 1 \mid R) \mid \pi_0\big] = \sum_R \pi_0(R \mid q)P(o(d) = 1 \mid R). \quad (7.15)$$

Then we use $\lambda(d \mid \pi_1, \pi_2)$ to indicate the difference in observance probability under $\pi_1$ or $\pi_2$:

$$
\begin{aligned}
\lambda(d \mid \pi_1, \pi_2) &= \mathbb{E}_R \big[ P(o(d) = 1 \mid R) \mid \pi_1 \big] - \mathbb{E}_R \big[ P(o(d) = 1 \mid R) \mid \pi_2 \big] \\
&= \sum_R \theta_{\mathrm{rank}(d|R)} \big( \pi_1(R \mid q_i) - \pi_2(R \mid q_i) \big).
\end{aligned}
\tag{7.16}
$$

Then, the IPS estimate function is formulated as:

$$
x_i = f_{\mathrm{IPS}}(q_i, R_i, c_i) = \sum_{d:\rho(d|q_i)>0} \frac{c_i(d)}{\rho(d \mid q_i)} \lambda(d \mid \pi_1, \pi_2).
\tag{7.17}
$$

Each click is weighted inversely to its examination probability, but items with a zero probability: $\rho(d \mid q_i) = 0$ are excluded. We note that these items can never be clicked:

$$
\forall q, d \, (\rho(d \mid q) = 0 \rightarrow c(d) = 0).
\tag{7.18}
$$

Before we prove unbiasedness, we note that given $\rho(d \mid q_i) > 0$:

$$
\begin{aligned}
\mathbb{E}\left[ \frac{c(d)}{\rho(d \mid q)} \right] &= \frac{\sum_R \pi_0(R \mid q) \theta_{\mathrm{rank}(d|R)} \zeta_{d,q}}{\rho(d \mid q_i)} \\
&= \frac{\sum_R \pi_0(R \mid q) \theta_{\mathrm{rank}(d|R)}}{\sum_{R'} \pi_0(R' \mid q) \theta_{\mathrm{rank}(d|R')}} \zeta_{d,q} = \zeta_{d,q}.
\end{aligned}
\tag{7.19}
$$

This, in turn, can be used to prove unbiasedness:

$$
\begin{aligned}
\mathbb{E}[f_{\mathrm{IPS}}(q_i, R_i, c_i)] &= \sum_q P(q) \sum_{d:\rho(d \mid q_i)>0} \zeta_{d,q} \lambda(d \mid \pi_1, \pi_2) \\
&= \mathbb{E}[\mathrm{CTR}(\pi_1)] - \mathbb{E}[\mathrm{CTR}(\pi_2)] = \Delta(\pi_1, \pi_2).
\end{aligned}
\tag{7.20}
$$

This proof is only valid under the following requirement:

$$
\forall d, q \, (\zeta_{d,q} \lambda(d \mid \pi_1, \pi_2) > 0 \rightarrow \rho(d \mid q) > 0).
\tag{7.21}
$$

In practice, this means that the items in the top-$k$ of either $\pi_1$ or $\pi_2$ need to have a non-zero examination probability under $\pi_0$, i.e., they must have a chance to appear in the top-$k$ under $\pi_0$.

Besides Requirement 7.21 the IPS counterfactual evaluation method [57, 127] is completely indifferent to $\pi_0$ and hence we do not consider it to be an online method. In the next section, we will introduce an algorithm for choosing and updating $\pi_0$ during logging to minimize the variance of the estimator. By doing so we turn counterfactual evaluation into an online method.

## 7.4 Logging Policy Optimization for Variance Minimization

Next, we introduce a method aimed at finding a logging policy minimizes the variance of the estimates of the counterfactual estimator.

## 7.4.1 Minimizing variance

In Section 7.3.3, we have discussed counterfactual evaluation and established that it is unbiased as long as $\theta$ is known and the logging policy meets Requirement 7.21. The variance of $\Delta_{IPS}$ depends on the position bias $\theta$, the conditional click probabilities $\zeta$, and the logging policy $\pi_0$. In contrast to the user-dependent $\theta$ and $\zeta$, the way data is logged by $\pi_0$ is something one can have control over. The goal of our method is to find the optimal policy that minimizes variance while still meeting Requirement 7.21:

$$\pi_0^* = \underset{\pi_0 \colon \pi_0 \text{ meets Req. 7.21}}{\arg \min} \text{Var}\big(\hat{\Delta}_{IPS}^{\pi_0}\big), \tag{7.22}$$

where $\hat{\Delta}_{IPS}^{\pi_0}$ is the counterfactual estimator based on data logged using $\pi_0$.

To formulate the variance, we first note that it is an expectation over queries:

$$\text{Var}(\hat{\Delta}) = \sum_q P(q) \text{Var}(\hat{\Delta} \mid q). \tag{7.23}$$

To keep notation short, for the remainder of this section we will write: $\Delta = \Delta(\pi_1, \pi_2)$; $\theta_{d,R} = \theta_{\text{rank}(d|R)}$; $\zeta_d = \zeta_{d,q}$; $\lambda_d = \lambda(d \mid \pi_1, \pi_2)$; and $\rho_d = \rho(d \mid q, \pi_0)$. Next, we consider the probability of a click pattern $c$, this is simply a vector indicating a possible combination of clicked documents $c(d) = 1$ and not-clicked documents $c(d) = 0$:

$$\begin{aligned} P(c \mid q) &= \sum_R \pi_0(R \mid q) \prod_{d:c(d)=1} \theta_{d,R} \zeta_d \prod_{d:c(d)=0} (1 - \theta_{d,R}\zeta_d) \\ &= \sum_R \pi_0(R \mid q) P(c \mid R). \end{aligned} \tag{7.24}$$

Here, $\pi_0$ has some control over this probability: by deciding the distribution of displayed rankings it can make certain click patterns more or less frequent. The variance added per query is the squared error of every possible click pattern weighted by the probability of each pattern. Let $\sum_c$ sum over every possible click pattern:

$$\text{Var}(\hat{\Delta}_{IPS}^{\pi_0} \mid q) = \sum_c P(c \mid q) \Big( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \Big)^2. \tag{7.25}$$

It is unknown whether there is a closed-form solution for $\pi_0^*$. However, the variance function is differentiable. Taking the derivative reveals a trade-off between two potentially conflicting goals:

$$\frac{\delta}{\delta \pi_0} \text{Var}(\hat{\Delta}_{IPS}^{\pi_0} \mid q) = \sum_c \overbrace{\left[ \frac{\delta}{\delta \pi_0} P(c \mid q) \right] \Big( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \Big)^2}^{\text{minimize frequency of high-error click patterns}} \\ + P(c \mid q) \underbrace{\left[ \frac{\delta}{\delta \pi_0} \Big( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \Big)^2 \right]}_{\text{minimize error of frequent click patterns}}. \tag{7.26}$$

On the one hand, the derivative reduces the frequency of click patterns that result in high error samples, i.e., by updating $\pi_0$ so that these are less likely to occur. On the other hand, changing $\pi_0$ also affects the propensities $\rho_d$, i.e., if $\pi_0$ makes an item $d$ less likely to be examined, its corresponding value $\lambda_d/\rho_d$ becomes larger, which can lead to a higher error for related click patterns. The optimal policy has to balance: (i) avoiding showing rankings that lead to high-error click patterns; and (ii) avoiding minimizing propensity scores, which increases the errors of corresponding click patterns.

Our method applies stochastic gradient descent to optimize the logging policy w.r.t. the variance. There are two main difficulties with this approach: (i) the parameters $\theta$ and $\zeta$ are unknown a priori; and (ii) the gradients include summations over all possible rankings and all possible click patterns, both of which are computationally infeasible. In the following sections, we will detail how LogOpt solves both of these problems.

## 7.4.2 Bias & relevance estimation

In order to compute the gradient in Eq. 7.26, the parameters $\theta$ and $\zeta$ have to be known. LogOpt is based on the assumption that accurate estimates of $\theta$ and $\zeta$ suffice to find a near-optimal logging policy. We note that the counterfactual estimator only requires $\theta$ to be known for unbiasedness (see Section 7.3.3). Our approach is as follows. At given intervals during evaluation we use the available clicks to estimate $\theta$ and $\zeta$. Then we use the estimated $\hat{\theta}$ to get the current estimate $\hat{\Delta}_{IPS}(\mathcal{I}, \hat{\theta})$ (Eq. 7.17) and optimize w.r.t. the estimated variance (Eq. 7.25) based on $\hat{\theta}$, $\hat{\zeta}$, and $\hat{\Delta}_{IPS}(\mathcal{I}, \hat{\theta})$.

For estimating $\theta$ and $\zeta$ we use the existing EM approach by Wang et al. [128], because it works well in situations where few interactions are available and does not require randomization. We note that previous work has found randomization-based approaches to be more accurate for estimating $\theta$ [4, 30, 128]. However, they require multiple interactions per query and specific types of randomization in their results; by choosing the EM approach we avoid having these requirements.

## 7.4.3 Monte-Carlo-based derivatives

Both the variance (Eq. 7.25) and its gradient (Eq. 7.26), include a sum over all possible click patterns. Moreover, they also include the probability of a specific pattern $P(c \mid q)$ that is based on a sum over all possible rankings (Eq. 7.24). Clearly, these equations are infeasible to compute under any realistic time constraints. To solve this issue, we introduce gradient estimation based on Monte-Carlo sampling. Our approach is similar to that of Ma et al. [78], however, we are estimating gradients of variance instead of general performance.

First, we assume that policies place the documents in order of rank and the probability of placing an individual document at rank $x$ only depends on the previously placed documents. Let $R_{1:x-1}$ indicate the (incomplete) ranking from rank 1 up to rank $x$, then $\pi_0(d \mid R_{1:x-1}, q)$ indicates the probability that document $d$ is placed at rank $x$ given that the ranking up to $x$ is $R_{1:x-1}$. The probability of a ranking $R$ up to rank $k$ is thus:

$$\pi_0(R_{1:k} \mid q) = \prod_{x=1}^{k} \pi_0(R_x \mid R_{1:x-1}, q). \tag{7.27}$$

Let $K$ be the length of a complete ranking $R$, the gradient of the probability of a ranking w.r.t. a policy is:

$$\frac{\delta \pi_0(R \mid q)}{\delta \pi_0} = \sum_{x=1}^{K} \frac{\pi_0(R \mid q)}{\pi_0(R_x \mid R_{1:x}, q)} \left[ \frac{\delta \pi_0(R_x \mid R_{1:x-1}, q)}{\delta \pi_0} \right]. \quad (7.28)$$

The gradient of the propensity w.r.t. the policy (cf. Eq. 7.15) is:

$$\frac{\delta \rho(d \mid q)}{\delta \pi_0} = \sum_{k=1}^{K} \theta_k \sum_{R} \pi_0(R_{1:k-1} \mid q) \left( \left[ \frac{\delta \pi_0(d \mid R_{1:k-1}, q)}{\delta \pi_0} \right] \right.$$
$$\left. + \sum_{x=1}^{k-1} \frac{\pi_0(d \mid R_{1:k-1}, q)}{\pi_0(R_x \mid R_{1:x-1}, q)} \left[ \frac{\delta \pi_0(R_x \mid R_{1:x-1}, q)}{\delta \pi_0} \right] \right). \quad (7.29)$$

To avoid iterating over all rankings in the $\sum_R$ sum, we sample $M$ rankings: $R^m \sim \pi_0(R \mid q)$, and a click pattern on each ranking: $c^m \sim P(c \mid R^m)$. This enables us to make the following approximation:

$$\widehat{\rho\text{-grad}}(d) = \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} \theta_k \left( \left[ \frac{\delta \pi_0(d \mid R_{1:k-1}^m, q)}{\delta \pi_0} \right] \right.$$
$$\left. + \sum_{x=1}^{k-1} \frac{\pi_0(d \mid R_{1:k-1}^m, q)}{\pi_0(R_x^m \mid R_{1:x-1}^m, q)} \left[ \frac{\delta \pi_0(R_x^m \mid R_{1:x-1}^m, q)}{\delta \pi_0} \right] \right), \quad (7.30)$$

since $\frac{\delta \rho(d \mid q)}{\delta \pi_0} \approx \widehat{\rho\text{-grad}}(d, q)$. In turn, we can use this to approximate the second part of Eq. 7.26:

$$\widehat{\text{error-grad}}(c) = 2 \left( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right) \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d^2} \widehat{\rho\text{-grad}}(d). \quad (7.31)$$

We approximate the first part of Eq. 7.26 with:

$$\widehat{\text{freq-grad}}(R, c) =$$
$$\left( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right)^2 \sum_{x=1}^{K} \frac{1}{\pi_0(R_x \mid R_{1:x-1}, q)} \left[ \frac{\delta \pi_0(R_x \mid R_{1:x-1}, q)}{\delta \pi_0} \right]. \quad (7.32)$$

Together, they approximate the complete gradient (cf. Eq. 7.26):

$$\frac{\delta \text{Var}(\hat{\Delta}_{IPS}^{\pi_0} \mid q)}{\delta \pi_0} \approx \frac{1}{M} \sum_{m=1}^{M} \widehat{\text{freq-grad}}(R^m, c^m) + \widehat{\text{error-grad}}(c^m). \quad (7.33)$$

Therefore, we can approximate the gradient of the variance w.r.t. a logging policy $\pi_0$, based on rankings sampled from $\pi_0$ and our current estimated click model $\hat{\theta}, \hat{\zeta}$, while staying computationally feasible.[1]

---

[1] For a more detailed description see Appendix 7.B.

---

**Algorithm 7.1** Logging-Policy Optimization Algorithm (LogOpt)

---

1: **Input**: Historical interactions: $\mathcal{I}$; rankers to compare $\pi_1, \pi_2$.
2: $\hat{\theta}, \hat{\zeta} \leftarrow$ infer_click_model($\mathcal{I}$)        *// estimate bias using EM*
3: $\hat{\lambda} \leftarrow$ estimated_observance($\hat{\theta}, \pi_1, \pi_2$)        *// estimate $\lambda$ cf. Eq 7.16*
4: $\hat{\Delta}(\pi_1, \pi_2) \leftarrow$ estimated_CTR($\mathcal{I}, \hat{\lambda}, \hat{\theta}$)        *// CTR diff. cf. Eq 7.17*
5: $\pi_0 \leftarrow$ init_policy()        *// initialize logging policy*
6: **for** $j \in \{1, 2, \dots\}$ **do**
7:     $q \sim P(q \mid \mathcal{I})$        *// sample a query from interactions*
8:     $\mathcal{R} \leftarrow \{R^1, R^2, \dots, R^M\} \sim \pi_0(R \mid q)$        *// sample M rankings*
9:     $\mathcal{C} \leftarrow \{c^1, c^2, \dots, c^M\} \sim P(c \mid \mathcal{R})$        *// sample M click patterns*
10:     $\hat{\delta} \leftarrow$ approx_grad($\mathcal{R}, \mathcal{C}, \hat{\lambda}, \hat{\theta}, \hat{\Delta}(\pi_1, \pi_2)$)        *// using Eq. 7.33*
11:     $\pi_0 \leftarrow$ update($\pi_0, \hat{\delta}$)        *// update using approx. gradient*
12: **return** $\pi_0$

---

## 7.4.4 Summary

We have summarized the LogOpt method in Algorithm 7.1. The algorithm requires a set of historical interactions $\mathcal{I}$ and two rankers $\pi_1$ and $\pi_2$ to compare. Then by fitting a click model on $\mathcal{I}$ using an EM-procedure (Line 2) an estimate of observation bias $\hat{\theta}$ and document relevance $\hat{\zeta}$ is obtained. Using $\hat{\theta}$, an estimate of the difference in observation probabilities $\hat{\lambda}$ is computed (Line 3 and cf. Eq 7.16), and an estimate of the CTR difference $\hat{\Delta}(\pi_1, \pi_2)$ (Line 4 and cf. Eq 7.17). Then the optimization of a new logging policy $\pi_0$ begins: A query is sampled from $\mathcal{I}$ (Line 7), and for that query $M$ rankings are sampled from the current $\pi_0$ (Line 8), then for each ranking a click pattern is sampled using $\hat{\theta}$ and $\hat{\zeta}$ (Line 9). Finally, using the sampled rankings and clicks, $\hat{\theta}$, $\hat{\lambda}$, and $\hat{\Delta}(\pi_1, \pi_2)$, the gradient is now approximated using Eq. 7.33 (Line 10) and the policy $\pi_0$ is updated accordingly (Line 11). This process can be repeated for a fixed number of steps, or until the policy has converged.

This concludes our introduction of LogOpt: the first method that optimizes the logging policy for faster convergence in counterfactual evaluation. We argue that LogOpt turns counterfactual evaluation into online evaluation, because it instructs which rankings should be displayed for the most efficient evaluation. The ability to make interventions like this is the defining characteristic of an online evaluation method.

# 7.5 Experimental Setup

We ran semi-synthetic experiments that are prevalent in online and counterfactual evaluation [41, 58, 86]. User-issued queries are simulated by sampling from learning to rank datasets; each dataset contains a preselected set of documents per query. We use the Yahoo! Webscope [17] and MSLR-WEB30k [95] datasets; they both contain 5-grade relevance judgements for all preselected query-document pairs. For each sampled query, we let the evaluation method decide which ranking to display and then simulate clicks on them using probabilistic click models.

To simulate position bias, we use the rank-based probabilities of Joachims et al.

---

[58]:

$$P(o(d) = 1 \mid R, q) = \frac{1}{\text{rank}(d \mid R)}. \tag{7.34}$$

If observed, the click probability is determined by the relevance label of the dataset (ranging from 0 to 4). More relevant items are more likely to be clicked, yet non-relevant documents still have a non-zero click probability:

$$P(c(d) = 1 \mid o(d) = 1, q) = 0.225 \cdot \text{relevance\_label}(q, d) + 0.1. \tag{7.35}$$

Spread over both datasets, we generated 2,000 rankers and created 1,000 ranker-pairs. We aimed to generate rankers that are likely to be compared in real-world scenarios; unfortunately, no simple distribution of such rankers is available. Therefore, we tried to generate rankers that have (at least) a decent CTR and that span a variety of ranking behaviors. Each ranker was optimized using LambdaLoss [129] based on the labelled data of 100 sampled queries; each ranker is based on a linear model that only uses a random sample of 50% of the dataset features. Figure 7.1 displays the resulting CTR distribution; it appears to follow a normal distribution, on both datasets.

For each ranker-pair and method, we sample $3 \cdot 10^6$ queries and calculate their CTR estimates for different numbers of queries. We considered three metrics: (i) The binary error: whether the estimate correctly predicts which ranker should be preferred. (ii) The absolute error: the absolute difference between the estimate and the true $\mathbb{E}[\text{CTR}]$ difference:

$$\text{absolute-error} = |\Delta(\pi_1, \pi_2) - \hat{\Delta}(\mathcal{I})|. \tag{7.36}$$

And (iii) the mean squared error: the squared error *per sample* (not the final estimate); if the estimator is unbiased this is equivalent to the empirical variance:

$$\text{mean-squared-error} = \frac{1}{N} \sum_{i=1}^{N} (\Delta(\pi_1, \pi_2) - x_i)^2. \tag{7.37}$$

We compare LogOpt with the following baselines: (i) A/B testing (with equal probabilities for each ranker), (ii) Team-Draft Interleaving, (iii) Probabilistic Interleaving (with $\tau = 4$), and (iv) Optimized Interleaving (with the inverse rank scoring function). Furthermore, we compare LogOpt with other choices of logging policies: (i) uniform sampling, (ii) A/B testing: showing either the ranking of A or B with equal probability, and (iii) an Oracle logging policy: applying LogOpt to the true relevances $\zeta$ and position bias $\theta$. We also consider LogOpt both in the case where $\theta$ is known *a priori*, or where it has to be estimated still. Because estimating $\theta$ and optimizing the logging policy $\pi_0$ is time-consuming, we only update $\hat{\theta}$ and $\pi_0$ after $10^3$, $10^4$, $10^5$ and $10^6$ queries. The policy LogOpt optimizes uses a neural network with 2 hidden layers consisting of 32 units each. The network computes a score for every document, then a softmax is applied to the scores to create a distribution over documents.

## 7.6  Results

Our results are displayed in Figures 7.2, 7.3, and 7.4. Figure 7.2 shows the results comparing LogOpt with other online evaluation methods; Figure 7.3 compares LogOpt
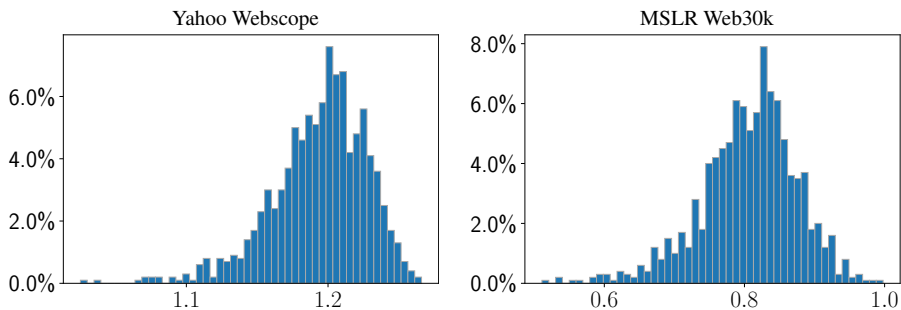
Figure 7.1: The CTR distribution of the 2000 generated rankers, 1000 were generated per dataset.

with counterfactual evaluation using other logging policies; and finally, Figure 7.4 shows the distribution of binary errors for each method after $3 \cdot 10^6$ sampled queries.

## 7.6.1 Performance of LogOpt

In Figure 7.2 we see that, unlike interleaving methods, counterfactual evaluation with LogOpt continues to decrease both its binary error and its absolute error as the number of queries increases. While interleaving methods converge at a binary error of at least 2.2% and an absolute error greater than $0.01$, LogOpt appears to converge towards zero errors for both. This is expected as LogOpt is proven to be unbiased when the position bias is known. Interestingly, we see similar behavior from LogOpt with estimated position bias. Both when bias is known or estimated, LogOpt has a lower error than the interleaving methods after $2 \cdot 10^3$ queries. Thus we conclude that interleaving methods converge faster and have an initial period where their error is lower, but are biased. In contrast, by being unbiased, LogOpt converges on a lower error eventually.

If we use Figure 7.2 to compare LogOpt with A/B testing, we see that on both datasets LogOpt has a considerably smaller mean squared error. Since both methods are unbiased, this means that LogOpt has a much lower variance and thus is expected to converge faster. On the Yahoo dataset we observe this behavior, both in terms of binary error and absolute error and regardless of whether the bias is estimated, LogOpt requires half as much data as A/B testing to reach the same level or error. Thus, on Yahoo LogOpt is roughly twice as data-efficient as A/B testing. On the MSLR dataset it is less clear whether LogOpt is noticeably more efficient: after $10^4$ queries the absolute error of LogOpt is twice as high, but after $10^5$ queries it has a lower error than A/B testing. We suspect that the relative drop in performance around $10^4$ queries is due to LogOpt overfitting on incorrect $\hat{\zeta}$ values, however, we were unable to confirm this. Hence, LogOpt is just as efficient as, or even more efficient than, A/B testing, depending on the circumstances.

Finally, when we use Figure 7.3 to compare LogOpt with other logging policy choices, we see that LogOpt mostly approximates the optimal Oracle logging policy. In contrast, the uniform logging policy is very data-inefficient; on both datasets it requires
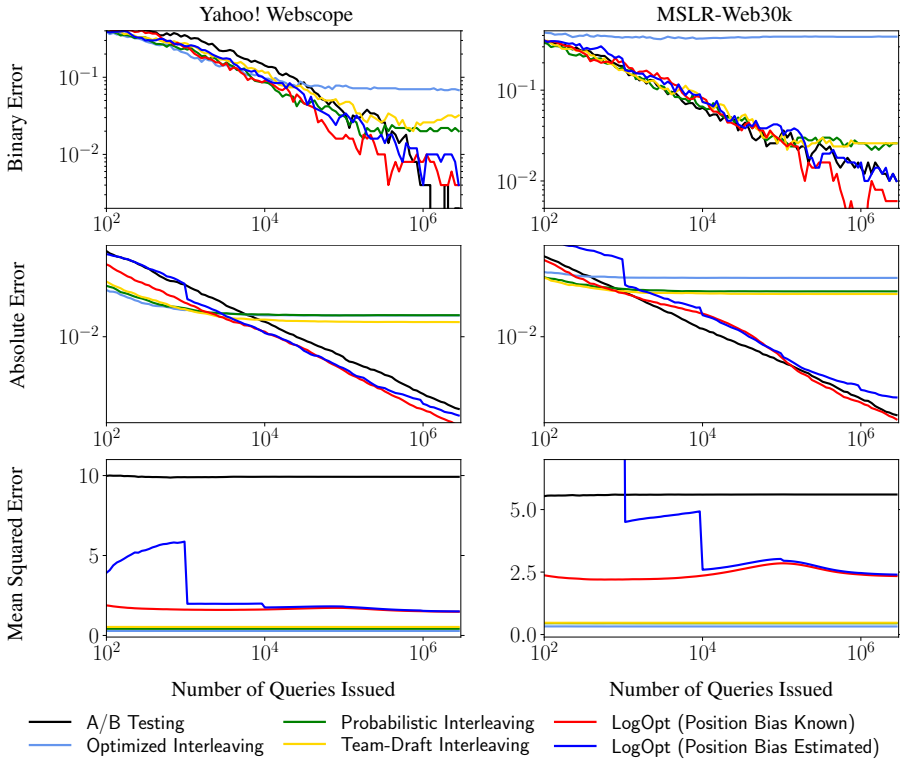
Figure 7.2: Comparison of LogOpt with other online methods; displayed results are an average over 500 comparisons.

around ten times the number of queries to reach the same level or error as LogOpt. The A/B logging policy is a better choice than the uniform logging policy, but apart from the dip in performance on the MSLR dataset, it appears to require twice as many queries as LogOpt. Interestingly, the performance of LogOpt is already near the Oracle when only $10^2$ queries have been issued. With such a small number of interactions, accurately estimating the relevances $\zeta$ should not be possible, thus it appears that in order for LogOpt to find an efficient logging policy the relevances $\zeta$ are not important. This must mean that only the differences in behavior between the rankers (i.e., $\lambda$) have to be known for LogOpt to be efficient. Overall, these results show that LogOpt can greatly increase the efficiency of counterfactual estimation.

## 7.6.2  Bias of interleaving

Our results in Figure 7.2 clearly illustrate the bias of interleaving methods: each of them systematically infers incorrect preferences in (at least) 2.2% of the ranker-pairs. These errors are systematic since increasing the number of queries from $10^5$ to $3 \cdot 10^6$ does not remove any of them. Additionally, the combination of the lowest mean-squared-error
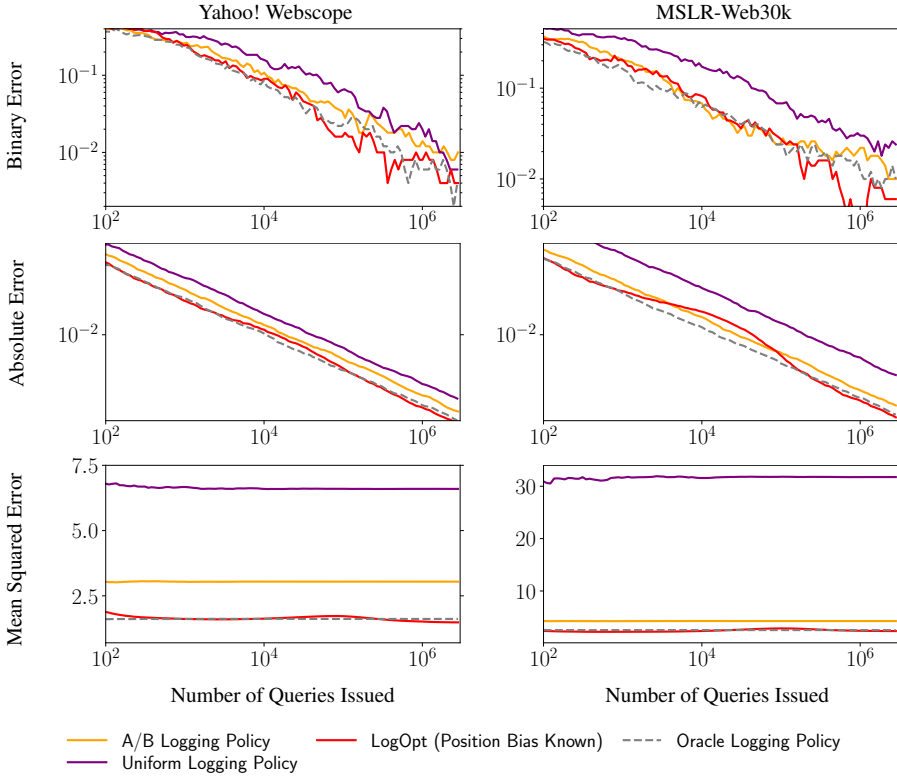
Figure 7.3: Comparison of logging policies for counterfactual evaluation; displayed results are an average over 500 comparisons.

with a worse absolute error than A/B testing after $10^4$ queries, indicates that interleaving results in a low variance at the cost of bias. To better understand when these systematic errors occur, we show the distribution of binary errors w.r.t. the CTR differences of the associated ranker-pairs in Figure 7.4. Here we see that most errors occur on ranker-pairs where the CTR difference is smaller than 1%, and that of all comparisons the percentage of errors greatly increases as the CTR difference decreases below 1%. This suggests that interleaving methods are unreliable to detect preferences when differences are 1% CTR or less.

It is hard to judge the impact this bias may have in practice. On the one hand, a 1% CTR difference is far from negligible: generally a 1% increase in CTR is considered an impactful improvement in the industry [102]. On the other hand, our results are based on a single click model with specific values for position bias and conditional click probabilities. While our results strongly prove that interleaving is biased, we should be careful not to generalize the size of the observed systematic error to all other ranking settings.

Previous work has performed empirical studies to evaluate various interleaving

methods with real users. Chapelle et al. [18] applied interleaving methods to compare ranking systems for three different search engines, and found team-draft interleaving to highly correlate with absolute measures such as CTR. However, we note that in the study by Chapelle et al. [18] no more than six rankers were compared, thus such a study would likely miss a systematic error of 2.2%. In fact, Chapelle et al. [18] note themselves that they cannot confidently claim team-draft interleaving is completely unbiased. Schuth et al. [110] performed a larger comparison involving 38 ranking systems, but again, too small to reliably detect a small systematic error.

It appears that the field is missing a large scale comparison that involves a large enough number of rankers to observe small systematic errors. If such an error is found, the next step is to identify if certain types of ranking behavior are erroneously and systematically disfavored. While these questions remain unanswered, we are concerned that the claims of unbiasedness in previous work on interleaving (see Section 7.3.2) give practitioners an unwarranted sense of reliability in interleaving.

## 7.7 Conclusion

In this chapter, we considered thesis research question **RQ7**: whether counterfactual evaluation methods for ranking can be extended to perform efficient and effective online evaluation. Our answer is positive: we have introduced the Logging-Policy Optimization Algorithm (LogOpt): the first method that optimizes a logging policy for minimal variance counterfactual evaluation. Counterfactual evaluation is proven to be unbiased w.r.t. position bias and item-selection bias under a wide range of logging policies. With the introduction of LogOpt, we now have an algorithm that can decide which rankings should be displayed for the fastest convergence. Therefore, we argue that LogOpt turns the IPS-based counterfactual evaluation approach – which is indifferent to the logging policy – into an online approach – which instructs the logging policy. Our experimental results show that LogOpt can lead to a better data-efficiency than A/B testing, while also showing that interleaving is biased.

This brings us to the second thesis research question that this chapter addressed, **RQ8**: whether interleaving methods are truly unbiased w.r.t. position bias. We answer this question negatively: Our experimental results clearly reveal a systematic error in interleaving, moreover, in Appendix 7.A we formally prove that cases exist where interleaving is affected by position bias. In other words, interleaving should not be considered unbiased under the most common definition of bias in counterfactual evaluation.

While our findings are mostly theoretical, they do suggest that future work should further investigate the bias in interleaving methods. Our results suggest that all interleaving methods make systematic errors, in particular when rankers with a similar CTR are compared. Furthermore, to the best of our knowledge, no empirical studies have been performed that could measure such a bias; our findings strongly show that such a study would be highly valuable to the field. Finally, LogOpt shows that in theory an evaluation method that is both unbiased and efficient is possible; if future work finds that these theoretical findings match empirical results with real users, this could be the start of a new line of theoretically-justified online evaluation methods.

Inspired by the success of this chapter to find a method effective at both online and counterfactual evaluation for ranking, Chapter 8 introduces a method that is effective at both online and counterfactual Learning to Rank (LTR). Together, these chapters show that the divide between online and counterfactual optimization/evaluation can be bridged.
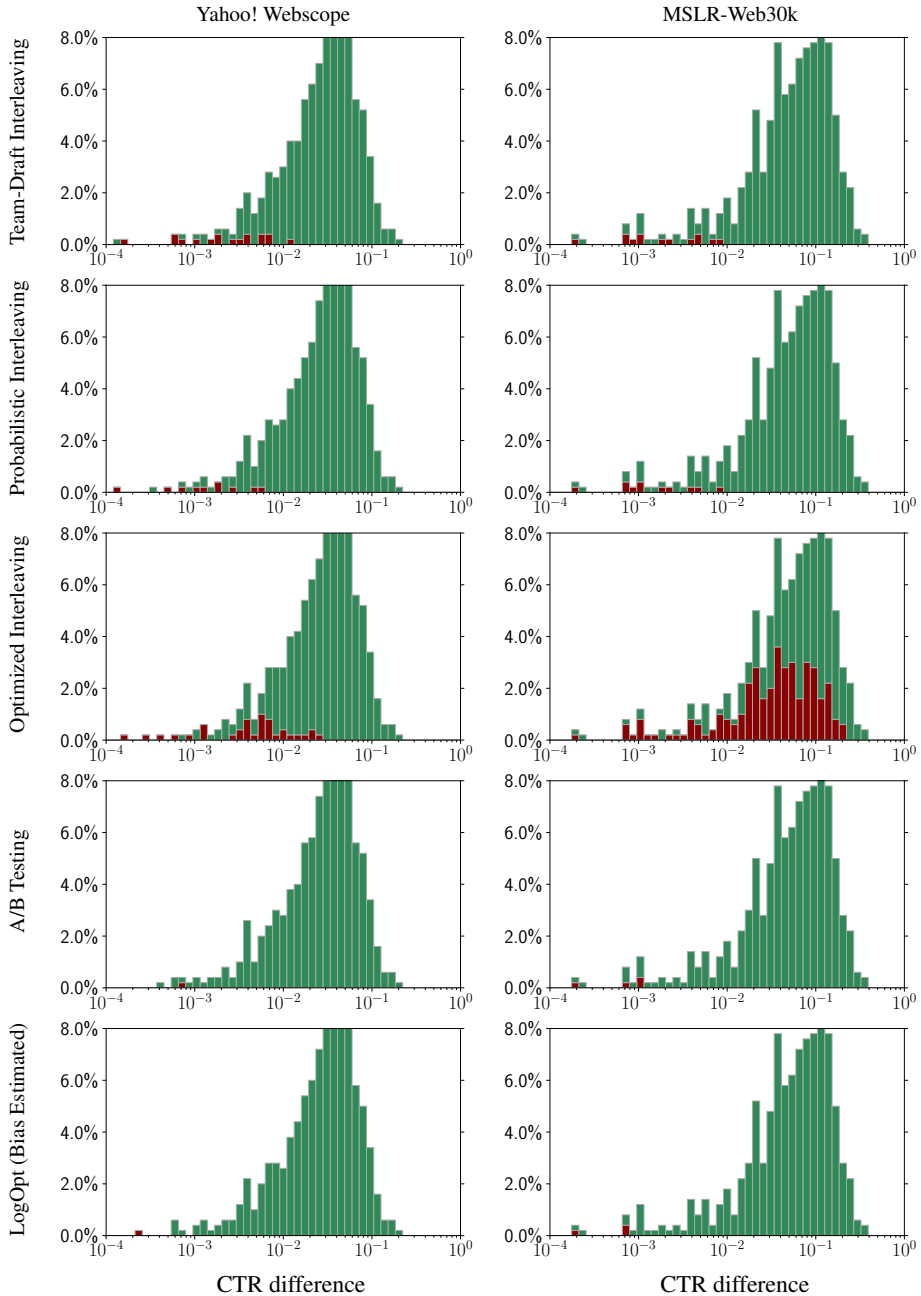
Figure 7.4: Distribution of errors over the CTR differences of the rankers in the comparison; red indicates a binary error; green indicates a correctly inferred binary preference; results are on estimates based on $3 \cdot 10^6$ sampled queries.

## 7.A   Proof of Bias in Interleaving

Section 7.3.2 claimed that for the discussed interleaving methods, an example can be constructed so that in expectation the wrong binary outcome is estimated w.r.t. the actual expected CTR differences. These examples are enough to prove that these interleaving methods are biased w.r.t. CTR differences. In the following sections we will introduce a single example for each interleaving method.

For clarity, we will keep these examples as basic as possible. We consider a ranking setting where only a single query $q_1$ occurs, i.e. $P(q_1) = 1$, furthermore, there are only three documents to be ranked: $A$, $B$, and $C$. The two policies $\pi_1$ and $\pi_2$ in the comparison are both deterministic so that: $\pi_1([A, B, C] \mid q_1) = 1$ and $\pi_2([B, C, A] \mid q_1) = 1$. Thus $\pi_1$ will always display the ranking: $[A, B, C]$, and $\pi_2$ the ranking: $[B, C, A]$. Furthermore, document $B$ is completely non-relevant: $\zeta_B = 0$, consequently, $B$ can never receive clicks; this will make our examples even simpler.

The true $\mathbb{E}[\text{CTR}]$ difference is thus:

$$\Delta(\pi_1, \pi_2) = (\theta_1 - \theta_3)\zeta_A + (\theta_3 - \theta_2)\zeta_C. \tag{7.38}$$

For each interleaving method, will now show that position bias parameters $\theta_1$, $\theta_2$, and $\theta_3$ and relevances $\zeta_A$ and $\zeta_C$ exist where the wrong binary outcome is estimated.

### 7.A.1   Team-Draft Interleaving

Team-Draft Interleaving [99] lets rankers take turns to add their top document and keeps track which ranker added each document. In total there are four possible interleaving and assignment combinations, each is equally probable:

| Interleaving | Ranking | Assignments | Probability |
|---|---|---|---|
| $R_1$ | A, B, C | 1, 2, 1 | 1/4 |
| $R_2$ | A, B, C | 1, 2, 2 | 1/4 |
| $R_3$ | B, A, C | 2, 1, 1 | 1/4 |
| $R_4$ | B, A, C | 2, 1, 2 | 1/4 |

Per issued query Team-Draft Interleaving produces a binary outcome, this is based on which ranker had most of its assigned documents clicked. To match our CTR estimate, we use 1 to indicate $\pi_1$ receiving more clicks, and $-1$ for $\pi_2$. Per interleaving we can

compute the probability of each outcome:

$$P(\text{outcome} = \ \ 1 \mid R_1) = \theta_1\zeta_A + (1 - \theta_1\zeta_A)\theta_3\zeta_C,$$
$$P(\text{outcome} = \ \ 1 \mid R_2) = \theta_1\zeta_A(1 - \theta_3\zeta_C),$$
$$P(\text{outcome} = \ \ 1 \mid R_3) = \theta_2\zeta_A + (1 - \theta_2\zeta_A)\theta_3\zeta_C,$$
$$P(\text{outcome} = \ \ 1 \mid R_4) = \theta_2\zeta_A(1 - \theta_3\zeta_C),$$
$$P(\text{outcome} = -1 \mid R_1) = 0,$$
$$P(\text{outcome} = -1 \mid R_2) = (1 - \theta_1\zeta_A)\theta_3\zeta_C,$$
$$P(\text{outcome} = -1 \mid R_3) = 0,$$
$$P(\text{outcome} = -1 \mid R_4) = (1 - \theta_2\zeta_A)\theta_3\zeta_C.$$

Since every interleaving is equally likely, we can easily derive the unconditional probabilities:

$$P(\text{outcome} = \ \ 1) = \frac{1}{4}\Big(\theta_1\zeta_A + (1 - \theta_1\zeta_A)\theta_3\zeta_C + \theta_1\zeta_A(1 - \theta_3\zeta_C)$$
$$+ \theta_2\zeta_A + (1 - \theta_2\zeta_A)\theta_3\zeta_C + \theta_2\zeta_A(1 - \theta_3\zeta_C)\Big),$$
$$P(\text{outcome} = -1) = \frac{1}{4}\Big((1 - \theta_1\zeta_A)\theta_3\zeta_C + (1 - \theta_2\zeta_A)\theta_3\zeta_C\Big).$$

With these probabilities, the expected outcome is straightforward to calculate:

$$\mathbb{E}[\text{outcome}] = P(\text{outcome} = 1) - P(\text{outcome} = -1)$$
$$= \frac{1}{4}\Big(\theta_1\zeta_A + \theta_1\zeta_A(1 - \theta_3\zeta_C) + \theta_2\zeta_A + \theta_2\zeta_A(1 - \theta_3\zeta_C)\Big) > 0.$$

Interestingly, without knowing the values for $\theta$, $\zeta_A$ and $\zeta_C$, we already know that the expected outcome is positive. Therefore, we can simply choose values that lead to a negative CTR difference, and the expected outcome will be incorrect. For this example, we choose the position bias: $\theta_1 = 1.0$, $\theta_2 = 0.9$, and $\theta_3 = 0.8$; and the relevances: $\zeta_1 = 0.1$, and $\zeta_3 = 1.0$. As a result, the expected binary outcome of Team-Draft Interleaving will not match the true $\mathbb{E}[\text{CTR}]$ difference:

$$\Delta(\pi_1, \pi_2) < 0 \wedge \mathbb{E}[\text{outcome}] > 0. \tag{7.39}$$

Therefore, we have proven that Team-Draft Interleaving is biased w.r.t. CTR differences.

## 7.A.2 Probabilistic Interleaving

Probabilistic Interleaving [41] treats rankings as distributions over documents, we follow the soft-max approach of Hofmann et al. [41] and use $\tau = 4.0$ as suggested. Probabilistic Interleaving creates interleavings by sampling randomly from one of the rankings, unlike Team-Draft Interleaving it does not remember which ranking added each document. Because rankings are treated as distributions, every possible permutation is a valid interleaving, leading to six possibilities with different probabilities of being displayed. When clicks are received, every possible assignment is considered and the expected outcome is computed over all possible assignments. Because there are 36 possible rankings and assignment combinations, we only report every possible ranking and the probabilities for documents $A$ or $C$ being added by $\pi_1$:

| Interleaving | Ranking | $P(\text{add}(A) = 1)$ | $P(\text{add}(C) = 1)$ | Probability |
|:---:|:---:|:---:|:---:|:---:|
| $R_1$ | A, B, C | 0.9878 | 0.4701 | 0.4182 |
| $R_2$ | A, C, B | 0.9878 | 0.4999 | 0.0527 |
| $R_3$ | B, A, C | 0.8569 | 0.0588 | 0.2849 |
| $R_4$ | B, C, A | 0.5000 | 0.0588 | 0.2094 |
| $R_5$ | C, A, B | 0.9872 | 0.5000 | 0.0166 |
| $R_6$ | C, B, A | 0.5000 | 0.0562 | 0.0182 |

These probabilities are enough to compute the expected outcome, similar as the procedure we used for Team-Draft Interleaving. We will not display the full calculation here as it is extremely long; we recommend using some form of computer assistance to perform these calculations. While there are many possibilities, we choose the following position bias: $\theta_1 = 1.0$, $\theta_2 = 0.9$, and $\theta_3 = 0.3$; and relevance: $\zeta_1 = 0.5$, and $\zeta_3 = 1.0$. This leads to the following erroneous result:

$$\Delta(\pi_1, \pi_2) < 0 \wedge \mathbb{E}[\text{outcome}] > 0. \tag{7.40}$$

Therefore, we have proven that Probabilistic Interleaving is biased w.r.t. CTR differences.

## 7.A.3 Optimized Interleaving

Optimized Interleaving casts interleaving as an optimization problem [96]. Optimized Interleaving works with a credit function: each clicked document produces a positive or negative credit. The sum of all credits is the final estimated outcome. We follow Radlinski and Craswell [96] and use the linear rank difference, resulting in the following credits per document: click-credit$(A) = 2$, click-credit$(B) = -1$, and click-credit$(C) = -1$. Then the set of allowed interleavings is created, these are all the rankings that do not contradict a pairwise document preference that both rankers agree on. Given this set of interleavings, a distribution over them is found so that if every document is equally relevant then no preference is found.[2] For our example, the only valid distribution over interleavings is the following:

| Interleaving | Ranking | Probability |
|:---:|:---:|:---:|
| $R_1$ | A, B, C | 1/3 |
| $R_2$ | B, A, C | 1/3 |
| $R_3$ | B, C, A | 1/3 |

The expected credit outcome shows us which ranker will be preferred in expectation:

$$\mathbb{E}[\text{credit}] = \frac{1}{3} \big( 2(\theta_1 + \theta_2 + \theta_3)\zeta_A - (\theta_2 + 2\theta_3)\zeta_C \big). \tag{7.41}$$

---

[2]Radlinski and Craswell [96] state that if clicks are not correlated with relevance then no preference should be found, in their click model (and ours) these two requirements are actually equivalent.

We choose the position bias: $\theta_1 = 1.0$, $\theta_2 = 0.9$, and $\theta_3 = 0.9$; and the relevances: $\zeta_1 = 0.5$, $\zeta_3 = 1.0$. As a result, the true $\mathbb{E}[\text{CTR}]$ difference is positive, but optimized interleaving will prefer $\pi_2$ in expectation:

$$\Delta(\pi_1, \pi_2) > 0 \wedge \mathbb{E}[\text{credit}] < 0. \tag{7.42}$$

Therefore, we have proven that Optimized Interleaving is biased w.r.t. CTR differences.

## 7.B   Expanded Explanation of Gradient Approximation

This section describes our Monte-Carlo approximation of the variance gradient in more detail. We repeat the steps described in Section 7.4.3 and include some additional intermediate steps; this should make it easier for a reader to verify our theory.

First, we assume that policies place the documents in order of rank and the probability of placing an individual document at rank $x$ only depends on the previously placed documents. Let $R_{1:x-1}$ indicate the (incomplete) ranking from rank 1 up to rank $x$, then $\pi_0(d \mid R_{1:x-1}, q)$ indicates the probability that document $d$ is placed at rank $x$ given that the ranking up to $x$ is $R_{1:x-1}$. The probability of a ranking $R$ of length $K$ is thus:

$$\pi_0(R \mid q) = \prod_{x=1}^{K} \pi_0(R_x \mid R_{1:x-1}, q). \tag{7.43}$$

The probability of a ranking $R$ up to rank $k$ is:

$$\pi_0(R_{1:k} \mid q) = \prod_{x=1}^{k} \pi_0(R_x \mid R_{1:x-1}, q). \tag{7.44}$$

Therefore the propensity (cf. Eq. 7.15) can be rewritten to:

$$\rho(d \mid q) = \sum_{k=1}^{K} \theta_k \sum_{R} \pi_0(R_{1:k-1} \mid q)\pi_0(d \mid R_{1:k-1}, q). \tag{7.45}$$

Before we take the gradient of the propensity, we note that the gradient of the probability of a single ranking is:

$$\frac{\delta\pi_0(R \mid q)}{\delta\pi_0} = \sum_{x=1}^{K} \frac{\pi_0(R \mid q)}{\pi_0(R_x \mid R_{1:x}, q)} \left[ \frac{\delta\pi_0(R_x \mid R_{1:x-1}, q)}{\delta\pi_0} \right]. \tag{7.46}$$

Using this gradient, we can derive the gradient of the propensity w.r.t. the policy:

$$\frac{\delta\rho(d \mid q)}{\delta\pi_0} = \sum_{k=1}^{K} \theta_k \sum_{R} \pi_0(R_{1:k-1} \mid q) \left( \left[ \frac{\delta\pi_0(d \mid R_{1:k-1}, q)}{\delta\pi_0} \right] \right.$$
$$\left. + \sum_{x=1}^{k-1} \frac{\pi_0(d \mid R_{1:k-1}, q)}{\pi_0(R_x \mid R_{1:x-1}, q)} \left[ \frac{\delta\pi_0(R_x \mid R_{1:x-1}, q)}{\delta\pi_0} \right] \right). \tag{7.47}$$

To avoid iterating over all rankings in the $\sum_R$ sum, we sample $M$ rankings: $R^m \sim \pi_0(R \mid q)$, and a click pattern on each ranking: $c^m \sim P(c \mid R^m)$. This enables us to make the following approximation:

$$\widehat{\rho\text{-grad}}(d) = \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} \theta_k \left( \left[ \frac{\delta \pi_0(d \mid R^m_{1:k-1}, q)}{\delta \pi_0} \right] \right.$$
$$\left. + \sum_{x=1}^{k-1} \frac{\pi_0(d \mid R^m_{1:k-1}, q)}{\pi_0(R^m_x \mid R^m_{1:x-1}, q)} \left[ \frac{\delta \pi_0(R^m_x \mid R^m_{1:x-1}, q)}{\delta \pi_0} \right] \right), \tag{7.48}$$

since $\frac{\delta \rho(d \mid q)}{\delta \pi_0} \approx \widehat{\rho\text{-grad}}(d, q)$. The second part of Eq. 7.26 is:

$$\left[ \frac{\delta}{\delta \pi_0} \left( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right)^2 \right] = 2 \left( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right) \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d^2} \left[ \frac{\delta \rho_d}{\delta \pi_0} \right], \tag{7.49}$$

using $\widehat{\rho\text{-grad}}(d)$ we get the approximation:

$$\widehat{\text{error-grad}}(c) = 2 \left( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right) \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d^2} \widehat{\rho\text{-grad}}(d). \tag{7.50}$$

Next, we consider the gradient of a single click pattern:

$$\frac{\delta}{\delta \pi_0} P(c \mid q) = \sum_R P(c \mid R) \left[ \frac{\delta \pi_0(R \mid q)}{\delta \pi_0} \right]. \tag{7.51}$$

This can then be used to reformulate the first part of Eq. 7.26:

$$\sum_c \left[ \frac{\delta}{\delta \pi_0} P(c \mid q) \right] \left( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right)^2$$
$$= \sum_c \sum_R P(c \mid R) \left[ \frac{\delta \pi_0(R \mid q)}{\delta \pi_0} \right] \left( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right)^2 \tag{7.52}$$

Making use of Eq. 7.46, we approximate this with:

$$\widehat{\text{freq-grad}}(R, c) =$$
$$\left( \Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right)^2 \sum_{x=1}^{K} \frac{1}{\pi_0(R_x \mid R_{1:x-1}, q)} \left[ \frac{\delta \pi_0(R_x \mid R_{1:x-1}, q)}{\delta \pi_0} \right]. \tag{7.53}$$

Combining the approximation of both parts of Eq. 7.26, allows us to approximate the complete gradient:

$$\frac{\delta \text{Var}(\hat{\Delta}_{IPS}^{\pi_0} \mid q)}{\delta \pi_0} \approx \frac{1}{M} \sum_{m=1}^{M} \widehat{\text{freq-grad}}(R^m, c^m) + \widehat{\text{error-grad}}(c^m). \tag{7.54}$$

This completes our expanded description of the gradient approximation. We have shown that we can approximate the gradient of the variance w.r.t. a logging policy $\pi_0$, based on rankings sampled from $\pi_0$ and our current estimated click model $\hat{\theta}$, $\hat{\zeta}$, while staying computationally feasible.

## 7.C   Notation Reference for Chapter 7

| Notation | Description |
|---|---|
| $k$ | the number of items that can be displayed in a single ranking |
| $i$ | an iteration number |
| $q$ | a user-issued query |
| $d$ | an item to be ranked |
| $R$ | a ranked list |
| $R_{1:x}$ | the subranking in $R$ from index 1 up to and including index $x$ |
| $\pi$ | a ranking policy |
| $\pi(R \mid q)$ | the probability that policy $\pi$ displays ranking $R$ for query $q$ |
| $\pi(R_x \mid R_{1:x-1}, q)$ | probability of $\pi$ adding item $R_x$ given $R_{1:x-1}$ is already placed |
| $\mathcal{I}$ | the available interaction data |
| $c$ | a click pattern: a vector indicating a combination of clicked and not-clicked items |
| $\sum_c$ | a summation over every possible click pattern |
| $c(d)$ | a function indicating item $d$ was clicked in click pattern $c$ |
| $o(d)$ | a function indicating item $d$ was observed at iteration $i$ |
| $x_i$ | the estimate for a single interaction $i$ |
| $f(q_i, R_i, c_i)$ | the method-specific function that converts a single interaction into an estimate $x_i$ |
| $\theta_{\text{rank}(d\mid R)}$ | the observation probability: $P(o(d) = 1 \mid R)$ |
| $\zeta_{d,q}$ | the conditional click probability: $P(c(d) = 1 \mid o(d) = 1, q)$ |

# 8

# Unifying Online and Counterfactual Learning to Rank

In Chapter 7, we introduced the Logging-Policy Optimization Algorithm (LogOpt) algorithm that turns a counterfactual ranking evaluation method into an online evaluation method. Thus, the contributions of Chapter 7 are a significant step in bridging the divide between online and counterfactual ranking evaluation. Inspired by this contribution, this chapter will consider whether something similar can be done for the gap between online and counterfactual Learning to Rank (LTR). Accordingly, in this chapter the following question will be addressed:

**RQ9** Can the counterfactual LTR approach be extended to perform highly effective online LTR?

In contrast with Chapter 7, which looked at finding the best logging policy, this chapter will consider a novel counterfactual estimator; we propose the novel *intervention-aware estimator* for both counterfactual and online LTR. The estimator corrects for the effect of position bias, trust bias, and item-selection bias using corrections based on the behavior of the logging policy and online interventions: changes to the logging policy made during the gathering of click data. Our experimental results show that, unlike existing counterfactual LTR methods, the intervention-aware estimator can greatly benefit from online interventions. In contrast, existing online methods are hindered without online interventions and thus should not be applied counterfactually. With the introduction of the intervention-aware estimator, we aim to bridge the online/counterfactual LTR division as it is shown to be highly effective in both online and counterfactual scenarios.

## 8.1   Introduction

Ranking systems form the basis for most search and recommendation applications [75]. As a result, the quality of such systems can greatly impact the user experience, thus it is important that the underlying ranking models perform well. The LTR field considers methods to optimize ranking models. Traditionally this was based on expert annotations. Over the years the limitations of expert annotations have become apparent; some of the

---

This chapter was submitted as [88]. Appendix 8.A gives a reference for the notation used in this chapter.

most important ones are: (i) they are expensive and time-consuming to acquire [17, 95]; (ii) in privacy-sensitive settings expert annotation is unethical, e.g., in email or private document search [128]; and (iii) often expert annotations appear to disagree with actual user preferences [104].

User interaction data solves some of the problems with expert annotations: (i) interaction data is virtually free for systems with active users; (ii) it does not require experts to look at potentially privacy-sensitive content; (iii) interaction data is indicative of users' preferences. For these reasons, interest in LTR methods that learn from user interactions has increased in recent years. However, user interactions are a form of implicit feedback and generally also affected by other factors than user preference [57]. Therefore, to be able to reliably learn from interaction data, the effect of factors other than preference has to be corrected for. In clicks on rankings three prevalent factors are well known: (i) *position bias*: users are less likely to examine, and thus click, lower ranked items [25]; (ii) *item-selection bias*: users cannot click on items that are not displayed [86, 92]; and (iii) *trust bias*: because users trust the ranking system, they are more likely to click on highly ranked items that they do not actually prefer [3, 57]. As a result of these biases, which ranking system was used to gather clicks can have a substantial impact on the clicks that will be observed. Current LTR methods that learn from clicks can be divided into two families: *counterfactual approaches* [58] – that learn from historical data, i.e., clicks that have been logged in the past – and *online approaches* [132] – that can perform interventions, i.e., they can decide what rankings will be shown to users. Recent work has noticed that some counterfactual methods can be applied as an online method [50], or vice versa [6, 136]. Nonetheless, every existing method was designed for either the online or counterfactual setting, never both.

In this chapter, we propose a novel estimator for both counterfactual and online LTR from clicks: the *intervention-aware estimator*. The intervention-aware estimator builds on ideas that underlie the latest existing counterfactual methods: the policy-aware estimator [86] and the affine estimator [123]; and expands them to consider the effect of online interventions. It does so by considering how the effect of bias is changed by an intervention, and utilizes these differences in its unbiased estimation. As a result, the intervention-aware estimator is both effective when applied as a counterfactual method, i.e., when learning from historical data, and as an online method where online interventions lead to enormous increases in efficiency. In our experimental results the intervention-aware estimator is shown to reach state-of-the-art LTR performance in both online and counterfactual settings, and it is the only method that reaches top-performance in both settings.

The main contributions of this chapter are:

1. A novel intervention-aware estimator that corrects for position bias, trust bias, item-selection bias, and the effect of online interventions.

2. An investigation into the effect of online interventions on state-of-the-art counterfactual and online LTR methods.

## 8.2   Interactions with Rankings

The theory in this chapter assumes that three forms of interaction bias occur: position bias, item-selection bias, and trust bias.

*Position bias* occurs because users only click an item after examining it, and users are more likely to examine items displayed at higher ranks [25]. Thus the rank (a.k.a. position) at which an item is displayed heavily affects the probability of it being clicked. We model this bias using $P(E = 1 \mid k)$: the probability that an item $d$ displayed at rank $k$ is examined by a user $E$ [128].

*Item-selection bias* occurs when some items have a zero probability of being examined in some displayed rankings [92]. This can happen because not all items are displayed to the user, or if the ranked list is so long that no user ever considers the entire list. We model this bias by stating:

$$\exists k, \forall k', \; (k' > k \rightarrow P(E = 1 \mid k') = 0), \tag{8.1}$$

i.e., there exists a rank $k$ such that items ranked lower than $k$ have no chance of being examined. The distinction between position bias and item-selection bias is important because some methods can only correct for the former if the latter is not present [86].

Finally, *trust bias* occurs because users trust the ranking system and, consequently, are more likely to perceive top ranked items as relevant even when they are not [57]. We model this bias using: $P(C = 1 \mid k, R, E)$: the probability of a click conditioned on the displayed rank $k$, the relevance of the item $R$, and examination $E$.

To combine these three forms of bias into a single click model, we follow Agarwal et al. [3] and write:

$$\begin{aligned}
P(C = 1 \mid d, k, q) \\
= P(E = 1 \mid k)\big(P(C = 1 \mid k, R = 0, E = 1)P(R = 0 \mid d, q) \\
+ P(C = 1 \mid k, R = 1, E = 1)P(R = 1 \mid d, q)\big),
\end{aligned} \tag{8.2}$$

where $P(R = 1 \mid d, q)$ is the probability that an item $d$ is deemed relevant w.r.t. query $q$ by the user. An analysis on real-world interaction data performed by Agarwal et al. [3], showed that this model better captures click behavior than models that only capture position bias [128] on search services for retrieving cloud-stored files and emails.

To simplify the notation, we follow Vardasbi et al. [123] and adopt:

$$\begin{aligned}
\alpha_k = P(E = 1 \mid k)\big(P(C = 1 \mid k, R = 1, E = 1) \\
- P(C = 1 \mid k, R = 0, E = 1)\big), \\
\beta_k = P(E = 1 \mid k)P(C = 1 \mid k, R = 0, E = 1).
\end{aligned} \tag{8.3}$$

This results in a compact notation for the click probability (8.2):

$$P(C = 1 \mid d, k, q) = \alpha_k P(R = 1 \mid d, q) + \beta_k. \tag{8.4}$$

For a single ranking $y$, let $k$ be the rank at which item $d$ is displayed in $y$; we denote $\alpha_k = \alpha_{d,y}$ and $\beta_k = \beta_{d,y}$. This allows us to specify the click probability conditioned on a ranking $y$:

$$P(C = 1 \mid d, y, q) = \alpha_{d,y} P(R = 1 \mid d, q) + \beta_{d,y}. \tag{8.5}$$

Finally, let $\pi$ be a ranking policy used for logging clicks, where $\pi(y \mid q)$ is the probability of $\pi$ displaying ranking $y$ for query $q$, then the click probability conditioned on $\pi$ is:

$$P(C = 1 \mid d, \pi, q) = \sum_y \pi(y \mid q)(\alpha_{d,y} P(R = 1 \mid d, q) + \beta_{d,y}). \qquad (8.6)$$

The proofs in the remainder of this chapter will assume this model of click behavior.

## 8.3 Background

In this section we cover the basics on LTR and counterfactual LTR.

### 8.3.1 Learning to rank

The field of LTR considers methods for optimizing ranking systems w.r.t. ranking metrics. Most ranking metrics are additive w.r.t. documents; let $P(q)$ be the probability that a user-issued query is query $q$, then the metric reward $\mathcal{R}$ commonly has the form:

$$\mathcal{R}(\pi) = \sum_q P(q) \sum_{d \in D_q} \lambda(d \mid D_q, \pi, q) P(R = 1 \mid d, q). \qquad (8.7)$$

Here, the $\lambda$ function scores each item $d$ depending on how $\pi$ ranks $d$ when given the preselected item set $D_q$; $\lambda$ can be chosen to match a desired metric, for instance, the common Discounted Cumulative Gain (DCG) metric [52]:

$$\lambda_{\text{DCG}}(d \mid D_q, \pi, q) = \sum_y \pi(y \mid q)(\log_2(\text{rank}(d \mid y) + 1))^{-1}. \qquad (8.8)$$

Supervised LTR methods can optimize $\pi$ to maximize $\mathcal{R}$ if relevances $P(R = 1 \mid d, q)$ are known [75, 129]. However in practice, finding these relevance values is not straightforward.

### 8.3.2 Counterfactual learning to rank

Over time, limitations of the supervised LTR approach have become apparent. Most importantly, finding accurate relevance values $P(R = 1 \mid d, q)$ has proved to be impossible or infeasible in many practical situations [127]. As a solution, LTR methods have been developed that learn from user interactions instead of relevance annotations. Counterfactual LTR concerns approaches that learn from historical interactions. Let $\mathcal{D}$ be a set of collected interaction data over $T$ timesteps; for each timestep $t$ it contains the user issued query $q_t$, the logging policy $\pi_t$ used to generate the displayed ranking $\bar{y}_t$, and the clicks $c_t$ received on the ranking:

$$\mathcal{D} = \{(\pi_t, q_t, \bar{y}_t, c_t)\}_{t=1}^T, \qquad (8.9)$$

where $c_t(d) \in \{0, 1\}$ indicates whether item $d$ was clicked at timestep $t$. While clicks are indicative of relevance they are also affected by several forms of bias, as discussed in Section 8.2.

Counterfactual LTR methods utilize estimators that correct for bias to unbiasedly estimate the reward of a policy $\pi$. The prevalent methods introduce a function $\hat{\Delta}$ that transforms a single click signal to correct for bias. The general estimate of the reward is:

$$\hat{\mathcal{R}}(\pi \mid \mathcal{D}) = \frac{1}{T} \sum_{t=1}^{T} \sum_{d \in D_{q_t}} \lambda(d \mid D_{q_t}, \pi, q) \hat{\Delta}(d \mid \pi_t, q_t, \bar{y}_t, c_t). \qquad (8.10)$$

We note the important distinction between the policy $\pi$ for which we estimate the reward, and the policy $\pi_t$ that was used to gather interactions at timestep $t$. During optimization only $\pi$ is changed in order to maximize the estimated reward.

The original Inverse Propensity Scoring (IPS) based estimator introduced by Wang et al. [127] and Joachims et al. [58] weights clicks according to examination probabilities:

$$\hat{\Delta}_{\text{IPS}}(d \mid \bar{y}_t, c_t) = \frac{c_t(d)}{P(E = 1 \mid \bar{y}_t, d)}. \qquad (8.11)$$

This estimator results in unbiased optimization under two requirements. First, every relevant item must have a non-zero examination probability in all displayed rankings:

$$\forall t, \forall d \in D_{q_t} \; (P(R = 1 \mid d, q_t) > 0 \rightarrow P(E = 1 \mid \bar{y}_t, d) > 0). \qquad (8.12)$$

Second, the click probability conditioned on relevance on examined items should be the same on every rank:

$$\forall k, k' \; (P(C \mid k, R, E = 1) = P(C \mid k', R, E = 1)), \qquad (8.13)$$

i.e., no trust bias is present. These requirements illustrate that this estimator can only correct for position bias, and is biased when item-selection bias or trust bias is present. For a proof we refer to previous work by Joachims et al. [58] and Vardasbi et al. [123].

Oosterhuis and de Rijke [86] (Chapter 5) adapt the IPS approach to correct for item-selection bias as well. They weight clicks according to examination probabilities conditioned on the logging policy, instead of the single displayed ranking on which a click took place. This results in the *policy-aware* estimator:

$$\begin{aligned} \hat{\Delta}_{\text{aware}}(d \mid \pi_t, q_t, c_t) &= \frac{c_t(d)}{P(E = 1 \mid \pi_t, q_t, d)} \\ &= \frac{c_t(d)}{\sum_y \pi(y \mid q_t) P(E = 1 \mid y, d, q_t)}. \end{aligned} \qquad (8.14)$$

This estimator can be used for unbiased optimization under two assumptions. First, every relevant item must have a non-zero examination probability under the logging policy:

$$\forall t, \forall d \in D_{q_t} \; (P(R = 1 \mid, d, q_t) > 0 \rightarrow P(E = 1 \mid \pi_t, d, q_t) > 0). \qquad (8.15)$$

Second, no trust bias is present as described in Eq. 8.13. Importantly, this first requirement can be met under item-selection bias, since a stochastic ranking policy can always provide every item a non-zero probability of appearing in a top-$k$ ranking. Thus, even

when not all items can be displayed at once, a stochastic policy can provide non-zero examination probabilities to all items. For a proof of this claim we refer to previous work by Oosterhuis and de Rijke [86].

Lastly, Vardasbi et al. [123] prove that IPS cannot correct for trust bias. As an alternative, they introduce an estimator based on affine corrections. This *affine* estimator penalizes an item displayed at rank $k$ by $\beta_k$ while also reweighting inversely w.r.t. $\alpha_k$:

$$\hat{\Delta}_{\text{affine}}(d \mid \bar{y}_t, c_t) = \frac{c_t(d) - \beta_{d,\bar{y}_t}}{\alpha_{d,\bar{y}_t}}. \tag{8.16}$$

The $\beta$ penalties correct for the number of clicks an item is expected to receive due to its displayed rank, instead of its relevance. The affine estimator is unbiased under a single assumption, namely that the click probability of every item must be correlated with its relevance in every displayed ranking:

$$\forall t, \forall d \in D_{q_t}, \, \alpha_{d,\bar{y}_t} \neq 0. \tag{8.17}$$

Thus, while this estimator can correct for position bias and trust bias, it cannot correct for item-selection bias. For a proof of these claims we refer to previous work by Vardasbi et al. [123].

We note that all of these estimators require knowledge of the position bias ($P(E = 1 \mid k)$) or trust bias ($\alpha$ and $\beta$). A lot of existing work has considered how these values can be inferred accurately [3, 30, 128]. The theory in this chapter assumes that these values are known.

This concludes our description of existing counterfactual estimators on which our method expands. To summarize, each of these estimators corrects for position bias, one also corrects for item-selection bias, and another also for trust bias. Currently, there is no estimator that corrects for all three forms of bias together.

## 8.4 Related Work

One of the earliest approaches to LTR from clicks was introduced by Joachims [54]. It infers pairwise preferences between items from click logs and uses pairwise LTR to update an SVM ranking model. While this approach had some success, in later work Joachims et al. [58] notes that position bias often incorrectly pushes the pairwise loss to flip the ranking displayed during logging. To avoid this biased behavior, Joachims et al. [58] proposed the idea of counterfactual LTR, in the spirit of earlier work by Wang et al. [127]. This led to estimators that correct for position bias using IPS weighting (see Section 8.3.2). This work sparked the field of counterfactual LTR which has focused on both capturing interaction biases and optimization methods that can correct for them. Methods for measuring position bias are based on EM optimization [128], a dual learning objective [5], or randomization [4, 30]; for trust bias only an EM-based approach is currently known [3]. Agarwal et al. [2] showed how counterfactual LTR can optimize neural networks and DCG-like methods through upper-bounding. Oosterhuis and de Rijke [86] introduced an IPS estimator that can correct for item-selection bias (see Section 8.3.2 and Chapter 5), while also showing that the LambdaLoss framework [129]

can be applied to counterfactual LTR (see Chapter 5). Lastly, Vardasbi et al. [123] proved that IPS estimators cannot correct for trust bias and introduced an affine estimator that is capable of doing so (see Section 8.3.2). There is currently no known estimator that can correct for position bias, item selection bias, and trust bias simultaneously.

The other paradigm for LTR from clicks is online LTR [132]. The earliest method, Dueling Bandit Gradient Descent (DBGD), samples variations of a ranking model and compares them using online evaluation [41]; if an improvement is recognized the model is updated accordingly. Most online LTR methods have increased the data-efficiency of DBGD [43, 111, 126]; later work found that DBGD is not effective at optimizing neural models [82] (Chapter 3) and often fails to find the optimal linear-model even in ideal scenarios [84] (Chapter 4). To these limitations, alternative approaches for online LTR have been proposed. Pairwise Differentiable Gradient Descent (PDGD) takes a pairwise approach but weights pairs to correct for position bias [82] (Chapter 3). While PDGD was found to be very effective and robust to noise [50, 84] (Chapter 4), it can be proven that its gradient estimation is affected by position bias, thus we do not consider it to be unbiased. In contrast, Zhuang and Zuccon [136] introduced Counterfactual Online Learning to Rank (COLTR), which takes the DBGD approach but uses a form of counterfactual evaluation to compare candidate models. Despite making use of counterfactual estimation, Zhuang and Zuccon [136] propose the method solely for online LTR.

Interestingly, with COLTR the line between online and counterfactual LTR methods starts to blur. Recent work by Jagerman et al. [50] applied the original counterfactual approach [58] as an online method and found that it lead to improvements. Furthermore, Ai et al. [6] noted that with a small adaptation PDGD can be applied to historical data. Although this means that some existing methods can already be applied both online and counterfactually, no method has been found that is the most reliable choice in both scenarios.

## 8.5   An Estimator Oblivious to Online Interventions

Before we propose the main contribution of this chapter, the intervention-aware estimator, we will first introduce an estimator that simultaneously corrects for position bias, item-selection bias, and trust bias, without considering the effects of interventions. Subsequently, the resulting intervention-oblivious estimator will serve as a method to contrast the intervention-aware estimator with.

Section 8.3.2 described how the policy-aware estimator corrects for item-selection bias by taking into account the behavior of the logging policy used to gather clicks [86]. Furthermore, Section 8.3.2 also detailed how the affine estimator corrects for trust bias by applying an affine transformation to individual clicks [123]. We will now show that a single estimator can correct for both item-selection bias and trust bias simultaneously, by combining the approaches of both these existing estimators.

First we note the probability of a click conditioned on a single logging policy $\pi_t$

can be expressed as:

$$P(C = 1 \mid d, \pi_t, q) = \sum_{\bar{y}} \pi_t(\bar{y} \mid q)(\alpha_{d,\bar{y}} P(R = 1 \mid d, q) + \beta_{d,\bar{y}})$$
$$= \mathbb{E}_{\bar{y}}[\alpha_d \mid \pi_t, q] P(R = 1 \mid d, q) + \mathbb{E}_{\bar{y}}[\beta_d \mid \pi_t, q]. \tag{8.18}$$

where the expected values of $\alpha$ and $\beta$ conditioned on $\pi_t$ are:

$$\mathbb{E}_{\bar{y}}[\alpha_d \mid \pi_t, q] = \sum_{\bar{y}} \pi_t(\bar{y} \mid q)\alpha_{d,\bar{y}},$$
$$\mathbb{E}_{\bar{y}}[\beta_d \mid \pi_t, q] = \sum_{\bar{y}} \pi_t(\bar{y} \mid q)\beta_{d,\bar{y}}. \tag{8.19}$$

By reversing Eq. 8.18 the relevance probability can be obtained from the click probability. We introduce our *intervention-oblivious estimator*, which applies this transformation to correct for bias:

$$\hat{\Delta}_{\text{IO}}(d \mid q_t, c_t) = \frac{c_t(d) - \mathbb{E}_{\bar{y}}[\beta_d \mid \pi_t, q_t]}{\mathbb{E}_{\bar{y}}[\alpha_d \mid \pi_t, q_t]}. \tag{8.20}$$

The intervention-oblivious estimator brings together the policy-aware and affine estimators: on every click it applies an affine transformation based on the logging policy behavior. Unlike existing estimators, we can prove that the intervention-oblivious estimator is unbiased w.r.t. our assumed click model (Section 8.2).

**Theorem 8.1.** *The estimated reward $\hat{\mathcal{R}}$ (Eq. 8.10) using the intervention-oblivious estimator (Eq. 8.20) is unbiased w.r.t. the true reward $\mathcal{R}$ (Eq. 8.7) under two assumptions: (1) our click model (Eq. 8.5), and (2) the click probability on every item, conditioned on the logging policies per timestep $\pi_t$, is correlated with relevance:*

$$\forall t, \forall d \in D_{q_t}, \quad \mathbb{E}_{\bar{y}}[\alpha_d \mid \pi_t, q_t] \neq 0. \tag{8.21}$$

*Proof.* Using Eq. 8.18 and Eq. 8.21 the relevance probability can be derived from the click probability by:

$$P(R = 1 \mid d, q) = \frac{P(C = 1 \mid d, \pi_t, q) - \mathbb{E}_{\bar{y}}[\beta_d \mid \pi_t, q]}{\mathbb{E}_{\bar{y}}[\alpha_d \mid \pi_t, q]}. \tag{8.22}$$

Eq. 8.22 can be used to show that $\hat{\Delta}_{\text{IO}}$ is an unbiased indicator of relevance:

$$\mathbb{E}_{\bar{y},c}\big[\hat{\Delta}_{\text{IO}}(d \mid q_t, c_t) \mid \pi_t\big] = \mathbb{E}_{\bar{y},c}\left[\frac{c_t(d) - \mathbb{E}_{t,\bar{y}}[\beta_d \mid \pi_t, q_t]}{\mathbb{E}_{\bar{y}}[\alpha_d \mid \pi_t, q_t]} \mid \pi_t, q_t\right]$$
$$= \frac{\mathbb{E}_{\bar{y},c}[c_t(d) \mid \pi_t, q_t] - \mathbb{E}_{\bar{y}}[\beta_d \mid \pi_t, q_t]}{\mathbb{E}_{\bar{y}}[\alpha_d \mid \pi_t, q_t]}$$
$$= \frac{P(C = 1 \mid d, \pi_t, q_t) - \mathbb{E}_{\bar{y}}[\beta_d \mid \pi_t, q_t]}{\mathbb{E}_{\bar{y}}[\alpha_d \mid \pi_t, q_t]}$$
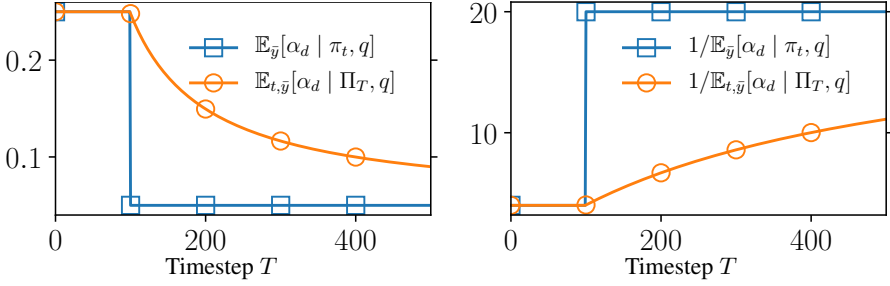$$= P(R = 1 \mid d, q_t). \tag{8.23}$$

Figure 8.1: Example of an online intervention and the weights used by the intervention-oblivious and intervention-aware estimators for a single item as more data is gathered.

Finally, combining Eq. 8.7 with Eq. 8.10 and Eq. 8.23 reveals that $\hat{\mathcal{R}}$ based on the intervention-oblivious estimator $\hat{\Delta}_{\text{IO}}$ is unbiased w.r.t. $\mathcal{R}$:

$$\mathbb{E}_{t,q,\bar{y},c}\Big[\hat{\mathcal{R}}(\pi \mid \mathcal{D})\Big] \tag{8.24}$$

$$= \sum_q P(q) \sum_{d \in D_q} \lambda(d \mid D_q, \pi, q) \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{\bar{y},c}\Big[\hat{\Delta}_{\text{IO}}(d \mid c, q) \mid \pi_t, q\Big]$$

$$= \sum_q P(q) \sum_{d \in D_q} \lambda(d \mid D_q, \pi, q) P(R = 1 \mid d, q) = \mathcal{R}(\pi). \qquad \square$$

### 8.5.1 Example with an online intervention

Existing estimators for counterfactual LTR are designed for a scenario where the logging policy is static:

$$\forall (\pi_t, \pi_{t'}) \in \mathcal{D}, \quad \pi_t = \pi_{t'}. \tag{8.25}$$

However, we note that if an online intervention takes place [50], meaning that the logging policy was updated during the gathering of data:

$$\exists (\pi_t, \pi_{t'}) \in \mathcal{D}, \quad \pi_t \neq \pi_{t'}, \tag{8.26}$$

the intervention-oblivious estimator is still unbiased. This was already proven in Theorem 8.1 because its assumptions cover both scenarios where online interventions do and do not take place.

However, the individual corrections of the intervention-oblivious estimator are only based on the single logging policy that was deployed at the timestep of each specific click. It is completely oblivious to the logging policies applied at different timesteps. Although this does not lead to bias in its estimation, it does result in unintuitive behavior. We illustrate this behavior in Figure 8.1, here a logging policy that results in $\mathbb{E}[\alpha_d \mid \pi_t, q] = 0.25$ for an item $d$ is deployed during the first $t \leq 100$ timesteps. Then an online intervention takes place and the logging policy is updated so that for $t > 100$, $\mathbb{E}[\alpha_d \mid \pi_t, q] = 0.05$. The intervention-oblivious estimator weights clicks inversely to $\mathbb{E}[\alpha_d \mid \pi_t]$; so clicks for $t \leq 100$ will be weighted by $1/0.25 = 4$ and

for $t > 100$ by $1/0.05 = 20$. Thus, there is a sharp and sudden difference in how clicks are treated before and after $t = 100$. What is unintuitive about this example is that the way clicks are treated after $t = 100$ is completely independent of what the situation was before $t = 100$. For instance, consider another item $d'$ where $\forall t, \mathbb{E}[\alpha_{d'} \mid \pi_t, q] = 0.05$. If both $d$ and $d'$ are clicked on timestep $t = 101$, these clicks would both be weighted by 20, despite the fact that $d$ has so far been treated completely different than $d'$. One would expect that in such a case the click on $d$ should be weighted less, to compensate for the high $\mathbb{E}[\alpha_d \mid \pi_t, q]$ it had in the first 100 timesteps. The question is whether such behavior can be incorporated in an estimator without introducing bias.

## 8.6 The Intervention-Aware Estimator

Our goal for the intervention-aware estimator is to find an estimator whose individual corrections are not only based on single logging policies, but instead consider the entire collection of logging policies used to gather the data $\mathcal{D}$. Importantly, this estimator should also be unbiased w.r.t. position bias, item-selection bias and trust bias.

For ease of notation, we use $\Pi_T$ for the set of policies that gathered the data in $\mathcal{D}$: $\Pi_T = \{\pi_1, \pi_2, \ldots, \pi_T\}$. The probability of a click can be conditioned on this set:

$$
\begin{aligned}
P(C = 1 \mid d, \Pi_T, q) &= \frac{1}{T} \sum_{t=1}^{T} \sum_{\bar{y}} \pi_t(\bar{y} \mid q)(\alpha_{d,\bar{y}} P(R = 1 \mid d, q) + \beta_{d,\bar{y}}) \\
&= \mathbb{E}_{t,\bar{y}}[\alpha_d \mid \Pi_T, q] P(R = 1 \mid d, q) + \mathbb{E}_{t,\bar{y}}[\beta_d \mid \Pi_T, q],
\end{aligned}
\tag{8.27}
$$

where the expected values of $\alpha$ and $\beta$ conditioned on $\Pi_T$ are:

$$
\begin{aligned}
\mathbb{E}_{t,\bar{y}}[\alpha_d \mid \Pi_T, q] &= \frac{1}{T} \sum_{t=1}^{T} \sum_{\bar{y}} \pi_t(\bar{y} \mid q) \alpha_{d,\bar{y}}, \\
\mathbb{E}_{t,\bar{y}}[\beta_d \mid \Pi_T, q] &= \frac{1}{T} \sum_{t=1}^{T} \sum_{\bar{y}} \pi_t(\bar{y} \mid q) \beta_{d,\bar{y}}.
\end{aligned}
\tag{8.28}
$$

Thus $P(C = 1 \mid d, \Pi_T, q)$ gives us the probability of a click given that any policy from $\Pi_T$ could be deployed. We propose our *intervention-aware estimator* that corrects for bias using the expectations conditioned on $\Pi_T$:

$$
\hat{\Delta}_{\text{IA}}(d \mid q_t, c_t) = \frac{c_t(d) - \mathbb{E}_{t,\bar{y}}[\beta_d \mid \Pi_T, q_t]}{\mathbb{E}_{t,\bar{y}}[\alpha_d \mid \Pi_T, q_t]}.
\tag{8.29}
$$

The salient difference with the intervention-oblivious estimator is that the expectations are conditioned on $\Pi_T$, all logging policies in $\mathcal{D}$, instead of an individual logging policy $\pi_t$. While the difference with the intervention-oblivious estimator seems small, our experimental results show that the differences in performance are actually quite sizeable. Lastly, we note that when no interventions take place the intervention-oblivious estimator and intervention-aware estimators are equivalent. Because the intervention-aware estimator is the only existing counterfactual LTR estimator whose corrections are

influenced by online interventions, we consider it to be a step that helps to bridge the gap between counterfactual and online LTR.

Before we revisit our online intervention example with our novel intervention-aware estimator, we prove that it is unbiased w.r.t. our assumed click model (Section 8.2).

**Theorem 8.2.** *The estimated reward $\hat{\mathcal{R}}$ (Eq. 8.10) using the intervention-aware estimator (Eq. 8.29) is unbiased w.r.t. the true reward $\mathcal{R}$ (Eq. 8.7) under two assumptions: (1) our click model (Eq. 8.5), and (2) the click probability on every item, conditioned on the set of logging policies $\Pi_T$, is correlated with relevance:*

$$\forall q, \forall d \in D_q, \quad \mathbb{E}_{t,\bar{y}}[\alpha_d \mid \Pi_T, q] \neq 0. \tag{8.30}$$

*Proof.* Using Eq. 8.27 and Eq. 8.30 the relevance probability can be derived from the click probability by:

$$P(R = 1 \mid d, q) = \frac{P(C = 1 \mid d, \Pi_T, q) - \mathbb{E}_{t,\bar{y}}[\beta_d \mid \Pi_T, q]}{\mathbb{E}_{t,\bar{y}}[\alpha_d \mid \Pi_T, q]}. \tag{8.31}$$

Eq. 8.31 can be used to show that $\hat{\Delta}_{\text{IA}}$ is an unbiased indicator of relevance:

$$
\begin{aligned}
\mathbb{E}_{t,\bar{y},c}\left[\hat{\Delta}_{\text{IA}}(d \mid q_t, c_t) \mid \Pi_T\right] &= \mathbb{E}_{t,\bar{y},c}\left[\frac{c_t(d) - \mathbb{E}_{t,\bar{y}}[\beta_d \mid \Pi_T, q_t]}{\mathbb{E}_{t,\bar{y}}[\alpha_d \mid \Pi_T, q_t]} \mid \Pi_T, q_t\right] \\
&= \frac{\mathbb{E}_{t,\bar{y},c}[c_t(d) \mid \Pi_T, q_t] - \mathbb{E}_{t,\bar{y}}[\beta_d \mid \Pi_T, q_t]}{\mathbb{E}_{t,\bar{y}}[\alpha_d \mid \Pi_T, q_t]} \\
&= \frac{P(C = 1 \mid d, \Pi_T, q_t) - \mathbb{E}_{t,\bar{y}}[\beta_d \mid \Pi_T, q_t]}{\mathbb{E}_{t,\bar{y}}[\alpha_d \mid \Pi_T, q_t]} \\
&= P(R = 1 \mid d, q_t).
\end{aligned}
\tag{8.32}
$$

Finally, combining Eq. 8.32 with Eq. 8.10 and Eq. 8.7 reveals that $\hat{\mathcal{R}}$ based on the intervention-aware estimator $\hat{\Delta}_{\text{IA}}$ is unbiased w.r.t. $\mathcal{R}$:

$$
\begin{aligned}
\mathbb{E}_{t,q,\bar{y},c}&\left[\hat{\mathcal{R}}(\pi \mid \mathcal{D})\right] \\
&= \sum_q P(q) \sum_{d \in D_q} \lambda(d \mid D_q, \pi, q) \mathbb{E}_{t,\bar{y},c}\left[\hat{\Delta}_{\text{IA}}(d \mid c, q) \mid \Pi_T, q\right] \\
&= \sum_q P(q) \sum_{d \in D_q} \lambda(d \mid D_q, \pi, q) P(R = 1 \mid d, q) \\
&= \mathcal{R}(\pi). \qquad \square
\end{aligned}
\tag{8.33}
$$

## 8.6.1  Online intervention example revisited

We will now revisit the example in Figure 8.1, but this time consider how the intervention-aware estimator treats item $d$. Unlike the intervention-oblivious estimator, clicks are weighted by $\mathbb{E}[\alpha_d \mid \Pi_T]$ which means that the exact timestep $t$ of a click does not matter, as long as $t < T$. Furthermore, the weight of a click can change as the total number of

timesteps $T$ increases. In other words, as more data is gathered, the intervention-aware estimator retroactively updates the weights of all clicks previously gathered.

We see that this behavior avoids the sharp difference in weights of clicks occurring before the intervention $t \leq 100$ and after $t > 100$. For instance, for a click on $d$ occuring at $t = 101$ while $T = 400$, results in $\mathbb{E}[\alpha_d \mid \Pi_T] = 0.1$ and thus a weight of $1/0.1 = 10$. This is much lower than the intervention-oblivious weight of $1/0.05 = 20$, because the intervention-aware estimator is also considering the initial period where $\mathbb{E}[\alpha_d \mid \pi_t, q]$ was high. Thus we see that the intervention-aware estimator has the behavior we intuitively expected: it weights clicks based on how the item was treated throughout all timesteps. In this example, it leads weights considerably smaller than those used by the intervention-oblivious estimator. In IPS estimators, low propensity weights are known to lead to high variance [58], thus we may expect that the intervention-aware estimator reduces variance in this example.

## 8.6.2   An online and counterfactual approach

While the intervention-aware estimator takes into account the effect of interventions, it does not prescribe what interventions should take place. In fact, it will work with any interventions that result in Eq. 8.30 being true, including the situation where no intervention takes place at all. For clarity, we will describe the intervention approach we applied during our experiments here. Algorithm 8.1 displays our online/counterfactual approach. As input it requires a starting policy ($\pi_0$), a choice for $\lambda$, the $\alpha$ and $\beta$ parameters, a set of intervention timesteps ($\Phi$), and the final timestep $T$.

The algorithm starts by initializing an empty set to store the gathered interaction data (Line 2) and initializes the logging policy with the provided starting policy $\pi_0$. Then for each timestep $i$ in $\Phi$ the dataset is expanded using the current logging policy so that $|\mathcal{D}| = i$ (Line 5). In other words, for $i - |\mathcal{D}|$ timesteps $\pi$ is used to display rankings to user-issued queries, and the resulting interactions are added to $\mathcal{D}$. Then a policy is optimized using the available data in $\mathcal{D}$ which becomes the new logging policy. For this optimization, we split the available data in training and validation partitions in order to do early stopping to prevent overfitting. We use stochastic gradient descent where we use $\pi_0$ as the initial model; this practice is based on the assumption that $\pi_0$ has a better performance than a randomly initialized model. Thus, during optimization, gradient calculation uses the intervention-aware estimator on the training partition of $\mathcal{D}$, and after each epoch, optimization is stopped if the intervention-aware estimator using the validation partition of $\mathcal{D}$ suspects overfitting. Each iteration results in an intervention as the resulting policy replaces the logging policy, and thus changes the way future data is logged. After iterating over $\Phi$ is completed, more data is gathered so that $|\mathcal{D}| = T$ and optimization is performed once more. The final policy is the end result of the procedure.

We note that, depending on $\Phi$, our approach can be either online, counterfactual, or somewhere in between. If $\Phi = \emptyset$ the approach is fully counterfactual since all data is gathered using the static $\pi_0$. Conversely, if $\Phi = \{1, 2, 3, \ldots, T\}$ it is fully online since at every timestep the logging policy is updated. In practice, we expect a fully online procedure to be infeasible as it is computationally expensive and user queries may be issued faster than optimization can be performed. In our experiments we will investigate the effect of the number of interventions on the approach's performance.

**Algorithm 8.1** Our Online/Counterfactual LTR Approach

1: **Input**: Starting policy: $\pi_0$; Metric weight function: $\lambda$;
         Inferred bias parameters: $\alpha$ and $\beta$;
         Interventions steps: $\Phi$; End-time: $T$.
2: $\mathcal{D} \leftarrow \{\}$                    *// initialize data container*
3: $\pi \leftarrow \pi_0$                    *// initialize logging policy*
4: **for** $i \in \Phi$ **do**
5:    $\mathcal{D} \leftarrow \mathcal{D} \cup \text{gather}(\pi, i - |\mathcal{D}|)$                    *// observe $i - |\mathcal{D}|$ timesteps*
6:    $\pi \leftarrow \text{optimize}(\mathcal{D}, \alpha, \beta, \pi_0)$                    *// optimize based on available data*
7: $\mathcal{D} \leftarrow \mathcal{D} \cup \text{gather}(\pi, T - |\mathcal{D}|)$                    *// expand data to T*
8: $\pi \leftarrow \text{optimize}(\mathcal{D}, \alpha, \beta, \pi_0)$                    *// optimize based on final data*
9: **return** $\pi$

## 8.7   Experimental Setup

Our experiments aim to answer the following research questions:

**RQ1** Does the intervention-aware estimator lead to higher performance than existing counterfactual LTR estimators when online interventions take place?

**RQ2** Does the intervention-aware estimator lead to performance comparable with existing online LTR methods?

We use the semi-synthetic experimental setup that is common in existing work on both online LTR [43, 82, 84, 136] and counterfactual LTR [58, 92, 123]. In this setup, queries and documents are sampled from a dataset based on commercial search logs, while user interactions and rankings are simulated using probabilistic click models. The advantage of this setup is that it allows us to investigate the effects of online interventions on a large scale while also being easy to reproduce by researchers without access to live ranking systems.

 We use the publicly-available Yahoo Webscope dataset [17], which consists of 29,921 queries with, on average, 24 documents preselected per query. Query-document pairs are represented by 700 features and five-grade relevance annotations ranging from not relevant (0) to perfectly relevant (4). The queries are divided into training, validation and test partitions.

 At each timestep, we simulate a user-issued query by uniformly sampling from the training and validation partitions. Subsequently, the preselected documents are ranked according to the logging policy, and user interactions are simulated on the top-5 of the ranking using a probabilistic click model. We apply Eq. 8.4 with $\alpha = [0.35, 0.53, 0.55, 0.54, 0.52]$ and $\beta = [0.65, 0.26, 0.15, 0.11, 0.08]$; the relevance probabilities are based on the annotations from the dataset: $P(R = 1 \mid d, q) = 0.25 \cdot$ relevance_label$(d, q)$. The values of $\alpha$ and $\beta$ were chosen based on those reported by Agarwal et al. [3] who inferred them from real-world user behavior. In doing so, we aim to emulate a setting where realistic levels of position bias, item-selection bias, and trust bias are present.

All counterfactual methods use the approach described in Section 8.6.2. To simulate a production ranker policy, we use supervised LTR to train a ranking model on 1% of the training partition [58]. The resulting production ranker has much better performance than a randomly initialized model, yet still leaves room for improvement. We use the production ranker as the initial logging policy. The size of $\Phi$ (the intervention timesteps) varies per run, and the timesteps in $\Phi$ are evenly spread on an exponential scale. All ranking models are neural networks with two hidden layers, each containing 32 hidden units with sigmoid activations. Gradients are calculated using a Monte-Carlo method following Oosterhuis and de Rijke [85] (Chapter 7). All policies apply a softmax to the document scores produced by the ranking models to obtain a probability distribution over documents. Clipping is only applied on the training clicks, denominators of any estimator are clipped by $10/\sqrt{|\mathcal{D}|}$ to reduce variance. Early stopping is applied based on counterfactual estimates of the loss using (unclipped) validation clicks.

The following methods are compared: (i) The intervention-aware estimator. (ii) The intervention-oblivious estimator. (iii) The policy-aware estimator [86] (Chapter 5). (iv) The affine estimator [123]. (v) PDGD [82] (Chapter 3), we apply PDGD both online and as a counterfactual method. As noted by Ai et al. [6], this can be done by separating the logging models from the learned model and, basing the debiasing weights on the logging function. (vi) Biased PDGD, identical to PDGD except that we do not apply the debiasing weights. (vii) COLTR [136]. We compute the Normalized DCG (NDCG) of both the logging policy and of a policy trained on all available data. Every reported result is the average of 20 independent runs, figures plot the mean, shaded areas indicate the standard deviation.

## 8.8 Results and Discussion

### 8.8.1 Comparison with counterfactual LTR

To answer the first research question: *whether the intervention-aware estimator leads to higher performance than existing counterfactual LTR estimators when online interventions take place*, we consider Figure 8.2 which displays the performance of LTR using different counterfactual estimators.

First we consider the top of Figure 8.2 which displays performance in the counterfactual setting where the logging policy is static.[1] We clearly see that the affine estimator converges at a suboptimal point of convergence, a strong indication of bias. The most probable cause is that the affine estimator is heavily affected by the presence of item-selection bias. In contrast, neither the policy-aware estimator nor the intervention-aware estimator have converged after $10^8$ queries. However, very clearly the intervention-aware estimator quickly reaches a higher performance. While the theory guarantees that it will converge at the optimal performance, we were unable to observe the number of queries it requires to do so. From the result in the counterfactual setting, we conclude that by correcting for position-bias, trust-bias, and item-selection bias the intervention-aware estimator already performs better without online interventions.

---

[1]Since under a static logging policy the intervention-aware and the intervention-oblivious estimators are equivalent, our conclusions apply to both in this setting.
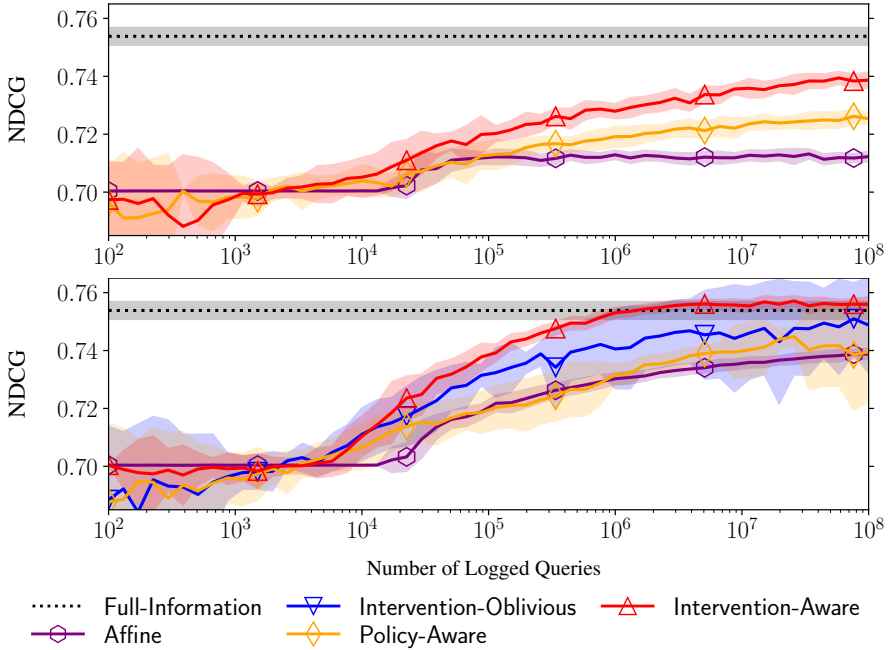
Figure 8.2: Comparison of counterfactual LTR estimators. Top: Counterfactual runs (no interventions); Bottom: Online runs (50 interventions).

Second, we turn to the bottom of Figure 8.2 which considers the online setting where the estimators perform 50 online interventions during logging. We see that online interventions have a positive effect on all estimators; leading to a higher performance for the affine and policy-aware estimators as well. However, interventions also introduce an enormous amount of variance for the policy-aware and intervention-oblivious estimators. In stark contrast, the amount of variance of the intervention-aware estimator hardly increases while it learns much faster than the other estimators.

Thus we answer the first research question positively: the intervention-aware estimator leads to higher performance than existing estimators, moreover, its data-efficiency becomes even greater when online interventions take place.

## 8.8.2 Effect of interventions

To better understand how much the intervention-aware estimator benefits from online interventions, we compared its performance under varying numbers of interventions in Figure 8.3. It shows both the performance of the resulting model when training from the logged data (top), as the performance of the logging policy which reveals when interventions take place (bottom). When comparing both graphs, we see that interventions lead to noticeable immediate improvements in data-efficiency. For instance, when only 5 interventions take place the intervention-aware estimator needs more than 20 times the amount of data to reach optimal performance as with 50 interventions. Despite these
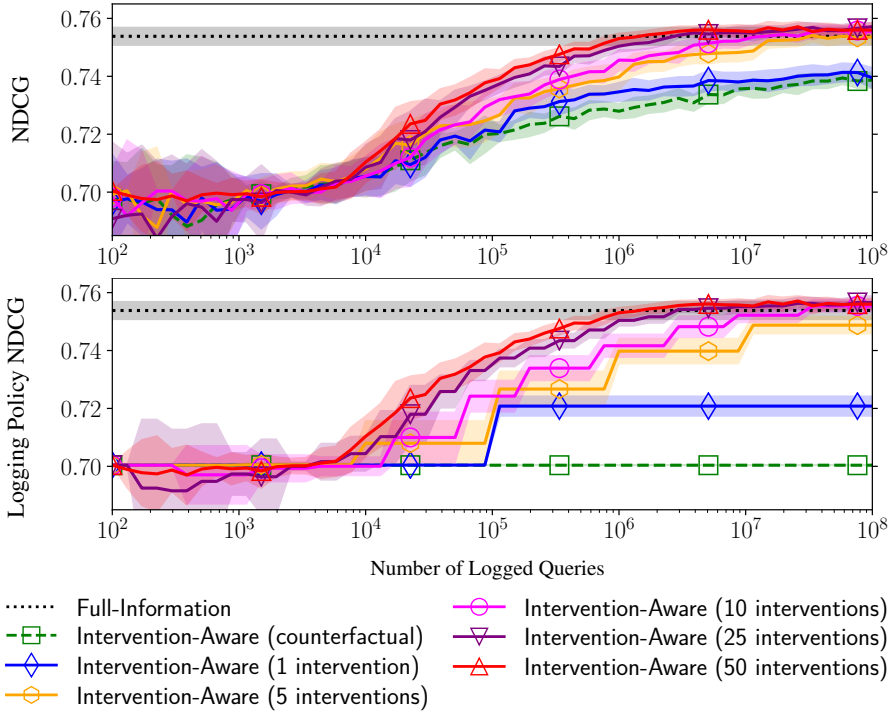
Figure 8.3: Effect of online interventions on LTR with the intervention-aware estimator.

speedups there are no large increases in variance. From these observations, we conclude that the intervention-aware estimator can effectively and reliably utilize the effect of online interventions for optimization, leading to enormous increases in data-efficiency.

### 8.8.3 Comparison with online LTR

In order to answer the second research question: *whether the intervention-aware estimator leads to performance comparable with existing online LTR methods*, we consider Figure 8.4 which displays the performance of two online LTR methods: PDGD and COLTR and the intervention-aware estimator with 100 online interventions.

First, we notice that COLTR is unable to outperform its initial policy, moreover, we see its performance drop as the number of iterations increase. We were unable to find hyper-parameters for COLTR where this did not occur. It seems likely that COLTR is unable to deal with trust-bias, thus causing this poor performance. However, we note that Zhuang and Zuccon [136] already show COLTR performs poorly when no bias or noise is present, suggesting that it is perhaps an unstable method overall.

Second, we see that the difference between PDGD and the intervention-aware estimator becomes negligible after $2 \cdot 10^4$ queries. Despite PDGD running fully online, and the intervention-aware estimator only performing 100 interventions in total. We do note that PDGD initially outperforms the intervention-aware estimator, thus it appears
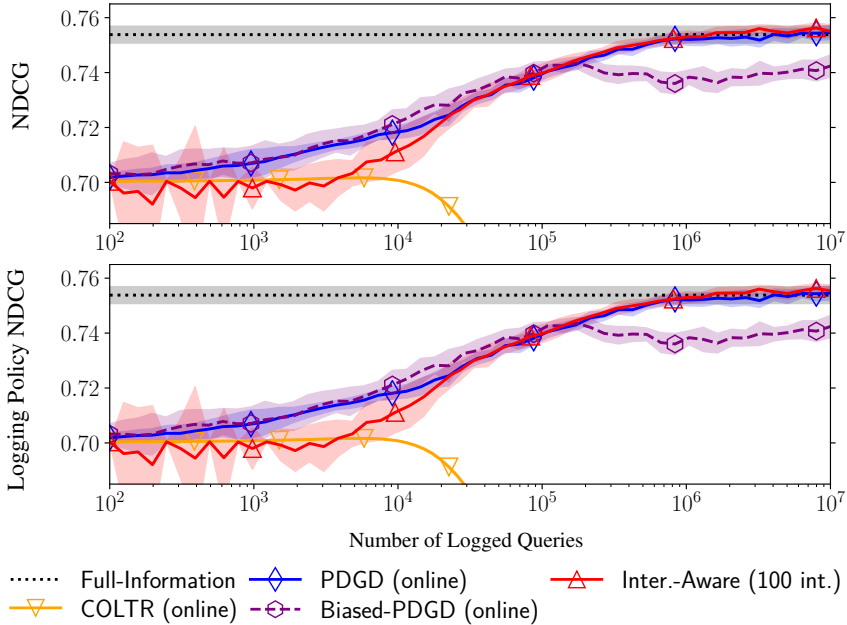
Figure 8.4: Comparison with online LTR methods.

that PDGD works better with low numbers of interactions. Additionally, we should also consider the difference in overhead: while PDGD requires an infrastructure that allows for fully online learning, the intervention-aware estimator only requires 100 moments of intervention, yet has comparable performance after a short initial period. By comparing Figure 8.4 to Figure 8.2, we see that the intervention-aware estimator is the first counterfactual LTR estimator that leads to stable performance while being comparably efficient with online LTR methods.

Thus we answer the second research question positively: besides an initial period of lower performance, the intervention-aware estimator has comparable performance to online LTR and only requires 100 online interventions to do so. To the best of our knowledge, it is the first counterfactual LTR method that can achieve this feat.

## 8.8.4 Understanding the effectiveness of PDGD

Now that we concluded that the intervention-aware estimator reaches performance comparable to PDGD when enough online interventions take place, the opposite question seems equally interesting: *Does PDGD applied counterfactually provide performance comparable to existing counterfactual LTR methods?*

To answer this question, we ran PDGD in a counterfactual way following Ai et al. [6], both fully counterfactual and with only 100 interventions. The results of these runs are displayed in Figure 8.5. Quite surprisingly, the performance of PDGD ran counterfactually and with 100 interventions, reaches much higher performance than the intervention-aware estimator without interventions. However, after a peak in perfor-
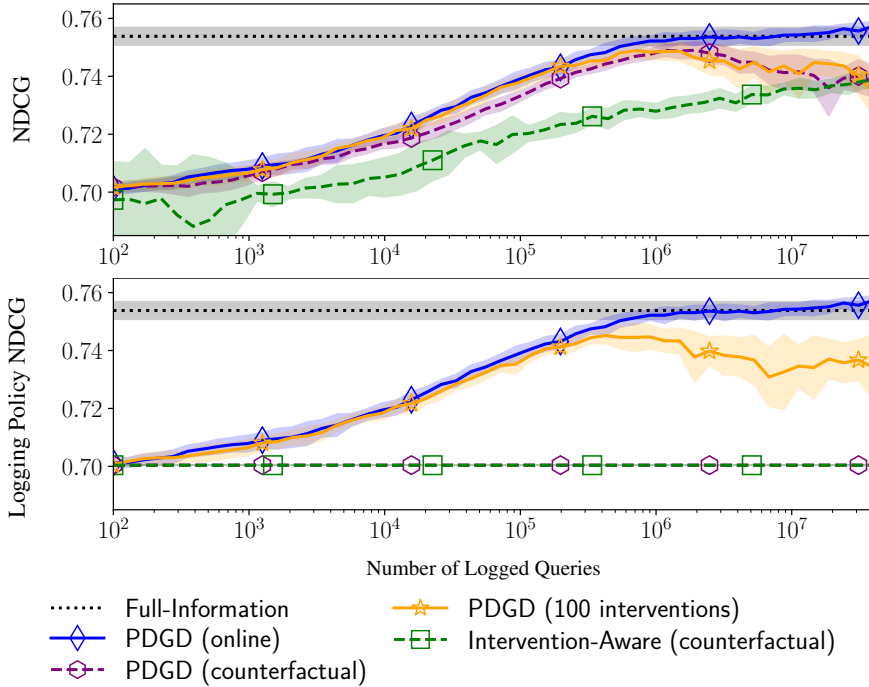
Figure 8.5: Effect of online interventions on PDGD.

mance around $10^6$ queries, the PDGD performance starts to drop. This drop cannot be attributed to overfitting, since online PDGD does not show the same behavior. Therefore, we must conclude that PDGD is biased when not ran fully online. This conclusion does not contradict the existing theory, since in Chapter 3 we only proved it is unbiased w.r.t. *pairwise* preferences. In other words, PDGD is not proven to unbiasedly optimize a ranking metric, and therefore also not proven to converge on the optimal model. This drop is particularly unsettling because PDGD is a continuous learning algorithm: there is no known early stopping method for PDGD. Yet these results show there is a great risk in running PDGD for too many iterations if it is not applied fully online. To answer our PDGD question: although PDGD reaches high performance when run counterfactually and appears to have great data-efficiency initially, it appears to converge at a suboptimal biased model. Thus we cannot conclude that PDGD is a reliable method for counterfactual LTR.

To better understand PDGD, we removed its debiasing weights resulting in the performance shown in Figure 8.4 (Biased-PDGD). Clearly, PDGD needs these weights to reach optimal performance. Similarly, from Figure 8.5 we see it also needs to be run fully online. This makes the choice between the intervention-aware estimator and PDGD complicated: on the one hand, PDGD does not require us to know the $\alpha$ and $\beta$ parameters, unlike the intervention-aware estimator; furthermore, PDGD has better initial data-efficiency even when not run fully online. On the other hand, there are no

theoretical guarantees for the convergence of PDGD, and we have observed that not running it fully online can lead to large drops in performance. It seems the choice ultimately depends on what guarantees a practitioner prefers.

## 8.9   Conclusion

In this chapter, we have introduced an intervention-aware estimator: an extension of existing counterfactual approaches that corrects for position-bias, trust-bias, and item-selection bias, while also considering the effect of online interventions. Our results show that the intervention-aware estimator outperforms existing counterfactual LTR estimators, and greatly benefits from online interventions in terms of data-efficiency. With only 100 interventions it is able to reach a performance comparable to state-of-the-art online LTR methods. These findings allow us to answer thesis research question **RQ9**: whether the counterfactual LTR approach be extended to perform highly effective online LTR. From our experimental results, it appears that the answer is positive: using the intervention-aware estimator and 100 online interventions the performance of state-of-the-art online LTR methods can be matched.

With the introduction of the intervention-aware estimator, we hope to further unify the fields of online LTR and counterfactual LTR as it appears to be the most reliable method for both settings. Future work could investigate what kind of interventions work best for the intervention-aware estimator. Since we have already seen in Chapter 7 that such an approach is effective for counterfactual/online ranking evaluation.

In retrospect, this chapter has put many findings from previous chapters in a different perspective. Chapter 3 introduced the concept of unbiasedness w.r.t. pairwise preferences and proved PDGD had this property. The experimental results of this chapter have shown that unbiasedness w.r.t. pairwise preferences is not enough to guarantee convergence at an optimal level of NDCG. Furthermore, Chapter 4 showed PDGD is very robust to noise and bias, but with the results of this chapter we now know that PDGD needs to be run online for this robustness. The policy-aware estimator in Chapter 5 is a precursor to the intervention-aware estimator of this chapter. While Chapter 5 realized that taking the logging policy into account is beneficial to counterfactual estimation, this chapter showed that taking the idea further, by accounting for all logging policies, provides even more benefits. Lastly, Chapter 7 looked at bridging the divide between online and counterfactual evaluation; in retrospect, the results of Chapter 7 might have been even better had it used the intervention-aware estimator. Together, Chapter 7 and this chapter suggest that an online method should both optimize its logging policy and use an intervention-aware estimator to learn, thus leaving a potentially very fruitful direction for future work.

## 8.A   Notation Reference for Chapter 8

| Notation | Description |
|----------|-------------|
| $k$ | the number of items that can be displayed in a single ranking |
| $t$ | a timestep number |
| $T$ | the total number of timesteps (gathered so far) |
| $\mathcal{D}$ | the available data |
| $\mathcal{R}(\pi)$ | the metric reward of a policy $\pi$ |
| $\hat{\mathcal{R}}(\pi \mid \mathcal{D})$ | an estimate of the metric reward of a policy $\pi$ |
| $q$ | a user-issued query |
| $D_q$ | the set of items to be ranked for query $q$ |
| $d$ | an item to be ranked |
| $y$ | a ranked list |
| $\pi$ | a ranking policy |
| $\pi(R \mid q)$ | the probability that policy $\pi$ displays ranking $R$ for query $q$ |
| $\pi(R_x \mid R_{1:x-1}, q)$ | probability of $\pi$ adding item $R_x$ given $R_{1:x-1}$ is already placed |
| $\Pi_T$ | the set of logging policies deployed up to timestep $T$ |
| $\lambda(d \mid D_q, \pi, q)$ | a metric function that weights items depending on their rank |
| $c(d)$ | a function indicating item $d$ was clicked in click pattern $c$ |
| $o(d)$ | a function indicating item $d$ was observed at iteration $i$ |

# 9
# Conclusions

In Section 1.1 we stated the overarching question that we aim to answer in this thesis:

> *Could there be a single general theoretically-grounded approach that has competitive performance for both evaluation and Learning to Rank (LTR) from user clicks on rankings, in both the counterfactual and online settings?*

The thesis has explored this question by looking at both online and counterfactual families of LTR methods, and in particular, to see if one of these approaches can be extended to be effective at both the online and counterfactual LTR scenarios. In this final chapter, we will summarize the findings of the thesis and discuss how they reflect on our overarching thesis question. Finally, we will consider future research directions for the field of LTR from user clicks.

## 9.1 Main Findings

This section look back at the thesis research questions posed in Section 1.1. We divide our discussion in two parts discussing online methods and counterfactual methods for LTR and evaluation, respectively.

### 9.1.1 Novel methods for online learning and evaluation for ranking

The first part of the thesis focussed on online LTR methods. Chapter 2 looked at multileaving methods [108] for comparing multiple ranking systems at once and asked:

**RQ1** Does the effectiveness of online ranking evaluation methods scale to large comparisons?

We introduced the novel Pairwise Preference Multileaving (PPM) algorithm, PPM bases evaluation on inferred pairwise item preferences. Furthermore, PPM is proven to have *fidelity* – it is provably unbiased in unambiguous cases [44] – and *considerateness* – it is safe w.r.t. the user experience during the gathering of clicks. From our theoretical analysis, we find that no other existing multileaving method manages to meet both criteria. In addition, our empirical results indicate that using PPM leads to a much lower

number of errors, in particular when applied to large scale comparisons. Therefore, we answered **RQ1** positively: PPM is shown to be effective at online ranking evaluation for large scale comparisons.

Besides in Chapter 2, online evaluation was also the subject of Chapter 7, which addressed the question:

**RQ8** Are existing interleaving methods truly capable of unbiased evaluation w.r.t. position bias?

We showed that under a basic rank-based model of position bias (common in counterfactual LTR [4, 58, 128]), three of the most prevalent interleaving algorithms are not unbiased: Team Draft Interleaving [99], Probabilistic Interleaving [41], and Optimized Interleaving [96]. For each of these three methods, we showed that situations exist where the binary outcome of the method does not agree with the expected binary difference in Click-Through-Rate (CTR). In other words, under a basic assumption of position bias, situations exist where these interleaving methods are expected to prefer one system over another, while the latter system has a higher expected CTR than the former. Thus, we answer **RQ8** negatively: the most prevalent interleaving methods are not unbiased w.r.t. position bias.

This finding can be extended to the multileaving methods: Team-Draft Multileaving [108], Probabilistic Multileaving [109], and Optimized Multileaving [108], since they are equivalent to their interleaving counterparts when only two systems are compared. While we did not examine it in this thesis, it is likely that PPM also fails to be unbiased under basic position bias. Nonetheless, an evaluation method can still be effective despite being biased, for instance, if the systematic error is small or situations where bias occurs are rare.

Chapter 3 looked at online LTR methods. Existing online LTR methods have relied on sampling model variants and comparing them using online evaluation [132]. In response to the existing online LTR approach, Chapter 3 considered the question:

**RQ2** Is online LTR possible without relying on model-sampling and online evaluation?

We answered this question positively by introducing Pairwise Differentiable Gradient Descent (PDGD), an online LTR method that learns from inferred pairwise preferences and uses a debiased pairwise loss. Besides proving that PDGD is unbiased w.r.t. pairwise preferences, our experimental results show that PDGD greatly outperforms the previous state-of-the-art Dueling Bandit Gradient Descent (DBGD) [132] algorithm in terms of data-efficiency and convergence. Furthermore, PDGD is the first online LTR method that can effectively optimize neural networks as ranking models.

Chapter 8 took another look at PDGD, in particular at conditions where PDGD is no longer effective. The results in Chapter 8 show that PDGD fails to reach optimal performance without debiasing weights or when not applied fully online. A particular worrisome observation was that, when not applied fully online, the performance of PDGD can degrade as more interactions are gathered. While this behavior looks similar, it is not overfitting since PDGD does not display it when applied online. Instead, it appears that PDGD becomes severely biased when not applied fully online. Therefore, we can conclude that the fact that PDGD is unbiased w.r.t. pairwise preferences is not

enough to guarantee unbiased optimization. It appears that we do not fully understand why PDGD appears to be so effective when run online.

The results of Chapter 3 had surprising implications on DBGD, for instance, it appeared that DBGD was not able to reach the performance of PDGD at convergence. Meanwhile, DBGD forms the basis of most existing online LTR methods. This prompted us to further investigate DBGD in Chapter 4, where we asked:

**RQ3** Are DBGD LTR methods reliable in terms of theoretical soundness and empirical performance?

By critically examining the theoretical assumptions underlying the DBGD method, we found that they are impossible when optimizing a deterministic ranking model. This means that the existing theoretical guarantees of DBGD are unsound in a lot of previous work where such models were used [40, 43, 82, 90, 111, 125, 126, 132, 135]. Moreover, our empirical analysis revealed that ideal circumstances exist where DBGD is still unable to find the optimal model. In other words, even in scenarios where optimization should be very easy, DBGD was unable to get near optimal performance. These findings lead us to answer **RQ3** negatively: our empirical results show that DBGD is very unreliable and its theoretical guarantees do not cover the most common LTR ranking models.

## 9.1.2 Extending the counterfactual approach to learning and evaluation for ranking

The second part of the thesis considered counterfactual LTR methods for optimization and evaluation. In particular, we tried to widen the applicability of counterfactual LTR methods and their effectiveness as online methods.

First, Chapter 5 recognized that the original Inverse Propensity Scoring (IPS) counterfactual method [58] is not unbiased when item selection bias occurs. This bias occurs when not all items can be displayed in a single ranking; this bias is unavoidable in top-$k$ ranking settings where only $k$ items can be displayed. One of the questions Chapter 5 addressed is:

**RQ4** Can counterfactual LTR be extended to top-$k$ ranking settings?

We showed that one can correct for item selection bias by basing propensity weights on both the position bias of the user and the stochastic ranking behavior of the logging policy. Our novel policy-aware estimator uses this idea to extend the original IPS approach by taking into account the logging policy behavior. We prove that, assuming rank-based position bias, the policy-aware estimator is unbiased as longs as the logging policy gives every relevant item a non-zero probability of appearing in the top-$k$ of a ranking. Furthermore, in our experimental results the policy-aware estimator approximates optimal performance regardless of the amount of item-selection bias present. Therefore, we answer **RQ4** positively: with the introduction of the policy-aware estimator the applicability of counterfactual LTR has been extended to top-$k$ ranking settings.

Besides learning from top-$k$ feedback, Chapter 5 also considered optimizing for top-$k$ metrics. Interestingly, the existing counterfactual LTR methods [2, 46] for optimizing

Discounted Cumulative Gain (DCG) metrics are very dissimilar from the state-of-the-art in supervised LTR [13, 129]. To address this dissimilarity, Chapter 5 posed the following question:

**RQ5** Is it possible to apply state-of-the-art supervised LTR methods to the counterfactual LTR problem?

We answer this question positively by showing that, with some small adjustments, the LambdaLoss framework [129] can be applied to counterfactual LTR losses, thus enabling the application of state-of-the-art supervised LTR to counterfactual LTR. The implication of this finding is that there does not need to be a division between state-of-the-art supervised LTR and counterfactual LTR. In other words, counterfactual LTR methods can build on the best methods from the supervised LTR field.

Chapter 6 takes a look at tabular and feature-based LTR methods. Tabular methods optimize a tabular ranking model [67–70, 139], which remembers the optimal ranking, in contrast with feature-based methods that optimize models that use the features of items to predict the optimal ranking. The tabular models are extremely expressive and can capture any possible ranking, making them always capable of converging on the optimal ranking [138]. However, their learned behavior does not generalize to previously unseen circumstances. Conversely, the learned behavior of feature-based models can generalize very well to previously unseen circumstances [10, 75]. But feature-based models can also be limited by the available features, because often the available features do not provide enough information to predict the optimal ranking. Thus feature-based LTR generalizes very well to unseen circumstances, whereas tabular LTR can specialize extremely well in specific circumstances. Inspired by this tradeoff, we asked the following question in Chapter 6:

**RQ6** Can the specialization ability of tabular online LTR be combined with the robust feature-based approach of counterfactual LTR?

Our answer is in the form of the novel Generalization and Specialization (GENSPEC) algorithm, a method for combining the behavior of a single robust generalized model and numerous specialized models. GENSPEC optimizes a single feature-based ranking model for performance across all queries, and many tabular ranking models each specialized for a single query. Then GENSPEC applies a meta-policy that uses high-confidence bounds to safely decide per query which model to deploy. Consequently, for previously unseen queries GENSPEC chooses the generalized model which utilizes robust feature-based prediction. For other queries, it can decide to deploy a specialized model, i.e., if it has enough data to confidently determine that the tabular model has found the better ranking. Our experimental results show that GENSPEC successfully combines robust performance on unseen queries with extremely high performance at convergence. Accordingly, we answer **RQ6** positively: using GENSPEC we can combine the specialization properties of tabular LTR with the robust generalization of feature-based LTR. For the LTR field, the introduction of GENSPEC shows that specialization does not need to be unique to tabular online LTR, instead it can be a property of counterfactual LTR as well.

As discussed above, Chapter 7 proved that several prominent interleaving methods are biased w.r.t. a basic model of position bias. Nonetheless, empirical results suggest

that these online ranking evaluation methods are still very effective. This leaves a gap for a theoretically-grounded online ranking evaluation method that is also very effective. To address this gap, Chapter 7 considers counterfactual ranking evaluation, which has strong theoretical guarantees, and asks:

**RQ7** Can counterfactual evaluation methods for ranking be extended to perform efficient and effective online evaluation?

We realized that with the introduction of the policy-aware estimator in Chapter 5, the logging policy has an important role in counterfactual estimation. Using the policy-aware estimator as a starting point, we introduce the Logging-Policy Optimization Algorithm (LogOpt) that optimizes the logging policy to minimize the variance of the policy-aware estimator. LogOpt can be deployed during the gathering of data, periodically or fully online, and thus changes the logging behavior through an intervention. As such, it turns the counterfactual evaluation approach with the policy-aware estimator into an online approach. Our experimental results show that applying LogOpt increases the data-efficiency of counterfactual evaluation with the policy-aware estimator. The performance with LogOpt is comparable to A/B testing and interleaving, but in contrast with interleaving, the policy-aware estimator applied with LogOpt does not have a systematic error. Therefore, we answer **RQ7** positively: by optimizing the logging policy with LogOpt, counterfactual evaluation can perform effective and data-efficient online evaluation.

Inspired by how Chapter 7 bridges part of the gap between online and counterfactual ranking evaluation, Chapter 8 addressed our final question:

**RQ9** Can the counterfactual LTR approach be extended to perform highly effective online LTR?

The motivation is similar to the previous chapter: we would like to find a theoretically-grounded method that is effective at both counterfactual LTR and online LTR. Since counterfactual LTR has strong theoretical guarantees, we used it as a starting point. Then we introduced the novel intervention-aware estimator which does not assume a stationary logging policy. As a result, the estimator takes into account the fact that an online intervention may change the logging policy during the gathering of data. Thus when applied online, the intervention-aware estimator does not only consider the logging policy used when a click was logged but also all the other logging policies applied at all other timesteps. In addition, the intervention-aware estimator also combines the theoretical properties of recent counterfactual LTR estimators: it is the first estimator that can correct for both position bias, item-selection bias, and trust bias. Our experimental results show that the intervention-aware estimator results in much lower variance, than an equivalent estimator that ignores the effect of interventions. Furthermore, in our experimental setting it outperformed all existing counterfactual estimators, with especially large differences when online interventions take place. Importantly, we observed that the intervention-aware estimator matches the performance of PDGD with as few as 100 interventions during learning. Besides a small initial period, LTR with the intervention-aware estimator was able to reach the performance of the most effective online LTR methods. Therefore, we answer **RQ9** positively: the intervention-aware estimator extends the counterfactual LTR approach to perform highly effective online

LTR. For the LTR field, this demonstrates that methods do not have to be either part of counterfactual LTR or online LTR. By designing them for both applications at once, they can be highly effective in both scenarios.

Finally, we note the complementary nature of the findings in the second part of the thesis. Many of the contributions of earlier chapters were used in later chapters. For instance, the methods introduced in Chapter 6 and Chapter 7 made use of the policy-aware estimator proposed in Chapter 5, and Chapter 8 built on the policy-aware estimator to introduce the intervention-aware estimator. Similarly, the adaptation of LambdaLoss for counterfactual LTR introduced in Chapter 5 was applied in the experiments of Chapter 6 and Chapter 7. While not explored in the thesis, many of the later contributions can also be applied to methods in earlier chapters. For instance, the intervention-aware estimator from Chapter 8 is completely compatible with the LambdaLoss adaption from Chapter 5 and GENSPEC from Chapter 6. In particular, it could be applied in combination with LogOpt from Chapter 7, potentially leading to even more effective online ranking evaluation. Together, the contributions of the second part can be combined into a single framework for counterfactual LTR and ranking evaluation, where our contributions complement each other. Importantly, this framework bridges several gaps between supervised LTR, online LTR, and counterfactual LTR.

## 9.2   Summary of Findings

The overarching question this thesis aimed to answer considered whether *there could be a single general theoretically-grounded approach that has competitive performance for both evaluation and LTR from user clicks on rankings, in both the counterfactual and online settings.*

We have looked at the family of online methods for LTR [43, 126, 132] and ranking evaluation [44, 56, 96, 108], which traditionally avoid making strong assumptions about user behavior, i.e., that a model of position bias is known [128]. While this makes their theory widely applicable, the theoretical guarantees of these methods are relatively weak. For instance, some interleaving and multileaving methods are proven to converge on correct outcomes if clicks are uncorrelated with relevance and thus every ranker performs equally well [41, 96]. Though such guarantees are valuable, they only cover a small group of unambiguous situations and thus leave most situations without theoretical guarantees. Online LTR methods are often motivated by empirical results from semi-synthetic experiments, where they are tested in settings with varying levels of noise and bias [42, 80, 111, 125]. The fundamental question with this type of empirical motivation is how well the results generalize, in particular, whether a method is still effective if the experimental conditions change slightly. This thesis has presented four examples of online methods that showed surprisingly poor performance when tested in new conditions: (i) On several datasets DBGD [132] did not get close to optimal performance after $10^6$ issued-queries while learning from clicks without noise or position bias (Chapter 4). (ii) Team Draft Interleaving [99], Probabilistic Interleaving [41], and Optimized Interleaving [96] make systematic errors in some ranking comparisons when tested under rank-based position bias (Chapter 7). (iii) The performance of the COLTR algorithm [136] dropped severely when tested

under position bias, item-selection bias, and trust bias (Chapter 8). (iv) PDGD no longer converged to near-optimal performance when we ran it counterfactually or only provided it with 100 online interventions, and instead resulted in a large drop in performance (Chapter 8). While these online methods LTR and evaluation have also shown great performance in previous work [41, 50, 96, 99, 111, 132, 136], these problematic examples illustrate why we cannot conclude that these online LTR methods are reliable. For instance, the performance of a method like PDGD was thought to be very robust to noise and bias [50] (Chapter 3 and 4), until tested without constant online interventions (Chapter 8). Without strong theoretical guarantees, we cannot know whether there are more currently-unknown conditions required for the robust performance of PDGD. In general, it is unclear how robust online LTR methods are in practice; this thesis has shown that there is a potential risk for detrimental performance if real-world circumstances do not match the tested experimental settings. Therefore, we conclude that online LTR methods should not be used as a basis for a single general approach for LTR and ranking evaluation from user clicks.

In the second part of the thesis, we considered the family of counterfactual methods for LTR and ranking evaluation [58, 127], which consist of theoretically-grounded methods that use explicit assumptions about user behavior. In contrast with the online family, counterfactual methods are less widely applicable: they only provide guarantees when the assumed models of user behavior are true. For instance, the original counterfactual LTR method assumes clicks are only affected by relevance and rank-based position bias [58, 127]. Despite their limited applicability, counterfactual methods have very strong theoretical guarantees. In contrast to most online LTR methods, counterfactual LTR methods guarantee convergence at the same performance as supervised LTR, given that their assumptions about user behavior are true. The findings of this thesis indicate that the strong guarantees with limited applicability of counterfactual LTR are preferable over the weak guarantees with wide applicability of online LTR. This is mainly because widening the applicability of counterfactual LTR proved very doable. In this thesis, we have expanded the applicability of counterfactual LTR and evaluation to (i) top-$k$ settings with item-selection bias (Chapter 5), and (ii) ranking settings where both trust bias and item-selection bias occur (Chapter 8). Besides expanding the settings where counterfactual LTR methods can be applied, we expanded the methods that perform counterfactual LTR, including: (iii) the state-of-the-art LambdaLoss supervised LTR framework [129] (Chapter 5), (iv) tabular models for extremely specialized rankings (Chapter 6), and (v) a meta-policy that safely chooses between generalized feature-based models and specialized tabular models (Chapter 6). Moreover, this thesis also found novel algorithms to increase the effectivity of counterfactual LTR methods for (vi) online ranking evaluation (Chapter 7), and (vii) online LTR (Chapter 8), even with a limited number of online interventions. Together, these contributions have widened the applicability of counterfactual LTR while maintaining its strong theoretical guarantees. As a direct result of this thesis, counterfactual LTR is applicable to more settings, more LTR methods can be applied to the counterfactual LTR problem, and counterfactual LTR methods are more effective in both the counterfactual and online LTR scenarios.

In conclusion, based on the findings of this thesis, it appears that counterfactual LTR could form the basis of a general approach for LTR from user clicks. In our experimental results, counterfactual LTR provided competitive performance to online LTR methods

in both the counterfactual and online settings. While the theory of counterfactual LTR does rely on stronger assumptions regarding user behavior than existing online LTR methods, counterfactual LTR provides far stronger theoretical guarantees. In contrast, it is currently unclear under what conditions online LTR methods are effective, making their performance very unpredictable. Therefore, we answer our overarching thesis question positively: the counterfactual LTR framework proposed in this thesis provides a unified approach for effective and reliable LTR from user clicks. For the LTR field, the counterfactual LTR framework bridges many gaps between areas of online LTR, counterfactual LTR, and supervised LTR, and as such, it unifies many of the most effective methods for LTR from user clicks.

## 9.3  Future Work

We will conclude the thesis with promising research directions for future work.

The most obvious direction is to widen the applicability of the counterfactual LTR framework. This means that estimators are introduced that are unbiased under other assumptions about user behavior. Joachims et al. [58] mentioned that the original counterfactual method is unbiased as long as click probabilities decompose into observation and relevance probabilities. For example, Vardasbi et al. [122] looked at the performance of counterfactual LTR when assuming cascading user behavior, an alternative to rank-based position bias. Additionally, Fang et al. [30] looked at context-dependent position bias, where the degree of bias varies per query. It seems natural to continue this trend to more complex models of user behavior. The challenge for future work is two-fold: find LTR methods that are proven to be unbiased under more complex user behavior models; and introduce methods that can reliably find the parameters of these behavior models.

Besides learning from more complex user behavior, there is a big need for LTR based on user clicks that optimizes for more complex goals. Some existing work has already looked at complex goals: for instance, Radlinski et al. [98] introduced a bandit algorithm for tabular LTR that optimizes for both relevance and diversity within a ranking. Thus, using user clicks to find a ranking that has relevant items, as well as having variety in the items within the ranking. Another example comes from Morik et al. [79], who use counterfactual LTR to optimize for relevance and ranking fairness. Ranking fairness metrics are based on the amount of exposure different items receive, for example, some fairness metrics measure whether certain groups of items receive similar amounts of exposure. Other areas of LTR also optimize for computational efficiency to ensure that ranking systems can process queries in minimal amounts of time [31]. Future work could investigate if counterfactual LTR can be used for complex goals like these and combinations of them.

Surprisingly, the experimental results in this thesis showed that PDGD is no longer effective when not applied fully online, and similarly, we observed very poor performance for the COLTR algorithm [136]. However, we could not find theoretically proven conditions that guarantee that PDGD or COLTR is or is not effective. It appears that we lack a theoretical approach to understand the limits of online LTR methods. If such an approach could be found, we may be able to correct for the faults in some online LTR

methods, or understand when they can be applied reliably. Thus it may be very valuable if future work reconsidered the theory behind existing online LTR methods.

Finally, most of the existing work on LTR from user interactions only consideres user clicks. Existing work has already looked at additional signals that are useful for learning [63, 110]. Novel methods that learn from other interactions in addition to user clicks have the potential to better understand user preferences. However, the main challenge this direction of research may be the availability of such data. Perhaps this direction of research mostly needs a publicly available source of data and methods to share such data in a privacy-respecting way.

Overall, our main advice for future work is to focus on methods that forge connections between advances in the larger field of LTR; that is, methods that combine the best of different areas, as our proposed framework does for online LTR, counterfactual LTR, and supervised LTR.

# Bibliography

[1] E. Adar, J. Teevan, S. T. Dumais, and J. L. Elsas. The web changes everything: Understanding the dynamics of web content. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 282–291, 2009. (Cited on pages 2, 38, 39, and 61.)

[2] A. Agarwal, K. Takatsu, I. Zaitsev, and T. Joachims. A general framework for counterfactual learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 5–14. ACM, 2019. (Cited on pages 3, 6, 78, 86, 92, 105, 109, 110, 156, and 173.)

[3] 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, 2019. (Cited on pages 3, 81, 88, 91, 94, 152, 153, 156, and 163.)

[4] 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. ACM, 2019. (Cited on pages 3, 78, 94, 131, 134, 156, and 172.)

[5] Q. Ai, K. Bi, C. Luo, J. Guo, and W. B. Croft. Unbiased learning to rank with unbiased propensity estimation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 385–394. ACM, 2018. (Cited on pages 62, 78, 81, 88, 89, 90, 91, 114, 131, and 156.)

[6] Q. Ai, T. Yang, H. Wang, and J. Mao. Unbiased learning to rank: Online or offline? *arXiv preprint arXiv:2004.13574*, 2020. (Cited on pages 152, 157, 164, and 167.)

[7] R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the world-wide web. *Nature*, 401(6749):130–131, 1999. (Cited on page 1.)

[8] J. Allan, B. Carterette, J. A. Aslam, V. Pavlu, B. Dachev, and E. Kanoulas. Million query track 2007 overview. In *TREC*. NIST, 2007. (Cited on pages 29 and 48.)

[9] W.-T. Balke, U. Güntzer, and W. Kießling. On real-time top $k$ querying for mobile services. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 125–143. Springer, 2002. (Cited on page 78.)

[10] C. M. Bishop. *Pattern Recognition and Machine Learning*, chapter 1.3. Springer, 2006. (Cited on pages 102 and 174.)

[11] A. Borisov, I. Markov, M. de Rijke, and P. Serdyukov. A neural click model for web search. In *WWW*, pages 531–541. International World Wide Web Conferences Steering Committee, 2016. (Cited on page 62.)

[12] B. Brost, I. J. Cox, Y. Seldin, and C. Lioma. An improved multileaving algorithm for online ranker evaluation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 745–748, 2016. (Cited on pages 4, 15, 16, 22, 23, 28, and 29.)

[13] C. J. Burges. From RankNet to LambdaRank to LambdaMART: An Overview. Technical report, Microsoft Research, 2010. (Cited on pages 1, 6, 41, 52, 79, 86, 87, 104, and 174.)

[14] F. Cai and M. de Rijke. A survey of query auto completion in information retrieval. *Foundations and Trends in Information Retrieval*, 10(4):273–363, 2016. (Cited on page 78.)

[15] 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. (Cited on page 42.)

[16] B. Carterette and P. Chandar. Offline comparative evaluation with incremental, minimally-invasive online feedback. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 705–714. ACM, 2018. (Cited on pages 3, 7, 78, 86, 89, and 94.)

[17] O. Chapelle and Y. Chang. Yahoo! Learning to Rank Challenge Overview. *Journal of Machine Learning Research*, 14:1–24, 2011. (Cited on pages 1, 2, 29, 38, 39, 48, 60, 61, 67, 89, 104, 109, 136, 152, and 163.)

[18] 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):1–41, 2012. (Cited on pages 1, 2, 17, 29, 126, and 141.)

[19] S. Chelaru, C. Orellana-Rodriguez, and I. S. Altingovde. How useful is social feedback for learning to rank YouTube videos? *World Wide Web*, 17(5):997–1025, 2014. (Cited on pages 1 and 38.)

[20] A. Chuklin, I. Markov, and M. de Rijke. *Click Models for Web Search*. Morgan & Claypool Publishers, 2015. (Cited on pages 2, 30, 40, 48, 60, 62, 68, and 127.)

[21] A. Chuklin, A. Schuth, K. Zhou, and M. D. Rijke. A comparative analysis of interleaving methods for

aggregated search. *ACM Transactions on Information Systems (TOIS)*, 33(2):1–38, 2015. (Cited on page 29.)

[22] G. Claeskens and N. L. Hjort. *Model Selection and Model Averaging*. Cambridge University Press, 2008. (Cited on page 102.)

[23] C. L. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2009 Web Track. In *TREC*. NIST, 2009. (Cited on page 29.)

[24] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu. Overview of the TREC 2003 Web Track. In *TREC*. NIST, 2003. (Cited on page 29.)

[25] 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. ACM, 2008. (Cited on pages 2, 104, 127, 152, and 153.)

[26] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010. (Cited on page 78.)

[27] D. Dato, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM Transactions on Information Systems (TOIS)*, 35(2):1–31, 2016. (Cited on pages 1, 29, 48, 67, 104, and 109.)

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. (Cited on page 87.)

[29] M. B. Dias, D. Locher, M. Li, W. El-Deredy, and P. J. Lisboa. The value of personalised recommender systems to e-business: a case study. In *Proceedings of the 2008 ACM conference on Recommender Systems*, pages 291–294, 2008. (Cited on page 1.)

[30] 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. (Cited on pages 134, 156, and 178.)

[31] L. Gallagher, R.-C. Chen, R. Blanco, and J. S. Culpepper. Joint optimization of cascade ranking models. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 15–23, 2019. (Cited on page 178.)

[32] S. C. Geyik, Q. Guo, B. Hu, C. Ozcaglar, K. Thakkar, X. Wu, and K. Kenthapadi. Talent search and recommendation systems at LinkedIn: Practical challenges and lessons learned. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1353–1354, 2018. (Cited on page 1.)

[33] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. (Cited on page 49.)

[34] C. A. Gomez-Uribe and N. Hunt. The Netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015. (Cited on page 1.)

[35] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *European Conference on Computer Vision*, pages 241–257. Springer, 2016. (Cited on page 1.)

[36] F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 124–131, 2009. (Cited on pages 29, 30, 47, 48, and 68.)

[37] D. M. Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004. (Cited on page 102.)

[38] J. He, C. Zhai, and X. Li. Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 2029–2032. ACM, 2009. (Cited on pages 1, 24, 48, and 67.)

[39] K. Hofmann. *Fast and Reliably Online Learning to Rank for Information Retrieval*. PhD thesis, University of Amsterdam, 2013. (Cited on pages 49 and 68.)

[40] K. Hofmann, S. Whiteson, and M. De Rijke. Balancing exploration and exploitation in learning to rank online. In *European Conference on Information Retrieval*, pages 251–263. Springer, 2011. (Cited on pages 40, 48, 49, 64, 67, 68, and 173.)

[41] 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, 2011. (Cited on pages 4, 17, 21, 22, 28, 29, 30, 40, 60, 62, 130, 131,

136, 145, 157, 172, 176, and 177.)

[42] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1):63–90, 2012. (Cited on pages 2, 5, 38, and 176.)

[43] 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. (Cited on pages 5, 40, 60, 62, 64, 94, 157, 163, 173, and 176.)

[44] K. Hofmann, S. Whiteson, and M. D. Rijke. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Transactions on Information Systems (TOIS)*, 31(4):1–43, 2013. (Cited on pages 2, 4, 16, 19, 20, 21, 22, 24, 28, 29, 126, 131, 171, and 176.)

[45] K. Hofmann, L. Li, and F. Radlinski. Online evaluation for information retrieval. *Foundations and Trends in Information Retrieval*, 10(1):1–117, 2016. (Cited on pages 1, 15, 17, and 126.)

[46] Z. Hu, Y. Wang, Q. Peng, and H. Li. Unbiased LambdaMART: An unbiased pairwise learning-to-rank algorithm. In *The World Wide Web Conference*, pages 2830–2836. ACM, 2019. (Cited on pages 3, 6, 88, and 173.)

[47] J. Huang, H. Oosterhuis, M. de Rijke, and H. van Hoof. Keeping dataset biases out of the simulation: A debiased simulator for reinforcement learning based recommender systems. In *Proceedings of the 2020 ACM conference on Recommender systems*, 2020. (Cited on page 12.)

[48] N. Hurley and M. Zhang. Novelty and diversity in top-n recommendation–analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)*, 10(4):14, 2011. (Cited on page 78.)

[49] R. Jagerman, H. Oosterhuis, and M. de Rijke. Query-level ranker specialization. In *CEUR Workshop Proceedings*, volume 2007, 2017. (Cited on page 12.)

[50] 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 *Proceedings of the 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 15–24. ACM, 2019. (Cited on pages 6, 7, 12, 89, 90, 94, 117, 152, 157, 159, and 177.)

[51] R. Jagerman, I. Markov, and M. de Rijke. Safe exploration for optimizing contextual bandits. *ACM Transactions on Information Systems*, 38(3):Article 24, 2020. (Cited on pages 106, 107, 108, 113, and 121.)

[52] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002. (Cited on pages 1 and 154.)

[53] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, volume 51, pages 243–250. ACM New York, NY, USA, 2017. (Cited on page 110.)

[54] 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, 2002. (Cited on pages 17, 19, 24, 38, 42, 61, 65, 79, 86, and 156.)

[55] T. Joachims. Unbiased evaluation of retrieval quality using clickthrough data. In *SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, volume 354, 2002. (Cited on pages 1 and 24.)

[56] T. Joachims. Evaluating retrieval performance using clickthrough data. In *Text Mining*. Physica Verlag, 2003. (Cited on pages 2, 4, 17, 126, 130, and 176.)

[57] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR Forum*, pages 154–161. ACM, 2005. (Cited on pages 78, 131, 132, 152, and 153.)

[58] 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, 2017. (Cited on pages 1, 2, 3, 4, 5, 6, 60, 62, 68, 78, 79, 80, 81, 86, 88, 89, 90, 91, 92, 103, 104, 105, 109, 110, 114, 120, 126, 128, 136, 137, 152, 155, 156, 157, 162, 163, 164, 172, 173, 177, and 178.)

[59] A. Karatzoglou, L. Baltrunas, and Y. Shi. Learning to rank for recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 493–494, 2013. (Cited on page 38.)

[60] S. K. Karmaker Santu, P. Sondhi, and C. Zhai. On application of learning to rank for e-commerce search. In *SIGIR*, pages 475–484. ACM, 2017. (Cited on pages 1 and 38.)

[61] S. Katariya, B. Kveton, C. Szepesvari, and Z. Wen. DCM bandits: Learning to rank with multiple clicks. In *International Conference on Machine Learning*, pages 1215–1224, 2016. (Cited on page 115.)

[62] A. Kazerouni, M. Ghavamzadeh, Y. A. Yadkori, and B. Van Roy. Conservative contextual linear bandits. In *Advances in Neural Information Processing Systems*, pages 3910–3919, 2017. (Cited on page 108.)

[63] E. Kharitonov, C. Macdonald, P. Serdyukov, and I. Ounis. Generalized team draft interleaving. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 773–782, 2015. (Cited on pages 18, 29, 33, and 179.)

[64] R. Kohavi and R. Longbotham. Online controlled experiments and A/B testing. *Encyclopedia of Machine Learning and Data Mining*, 7(8):922–929, 2017. (Cited on pages 126 and 129.)

[65] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, 2009. (Cited on page 17.)

[66] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann. Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1168–1176, 2013. (Cited on page 17.)

[67] J. Komiyama, J. Honda, and H. Nakagawa. Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1152–1161. JMLR.org, 2015. (Cited on pages 3, 6, 94, and 174.)

[68] 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. (Cited on pages 3, 103, and 115.)

[69] P. Lagrée, C. Vernade, and O. Cappé. Multiple-play bandits in the position-based model. In *Advances in Neural Information Processing Systems*, pages 1597–1605, 2016. (Cited on pages 3, 94, 110, 115, and 116.)

[70] T. Lattimore and C. Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020. (Cited on pages 3, 6, 102, 103, and 174.)

[71] D. Lefortier, P. Serdyukov, and M. De Rijke. Online exploration for detecting shifts in fresh intent. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 589–598, 2014. (Cited on pages 2, 38, 39, and 61.)

[72] S. Li, Y. Abbasi-Yadkori, B. Kveton, S. Muthukrishnan, V. Vinay, and Z. Wen. Offline evaluation of ranking policies with click models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1685–1694. ACM, 2018. (Cited on pages 94 and 95.)

[73] Z. Li, A. Grotov, J. Kiseleva, M. de Rijke, and H. Oosterhuis. Optimizing interactive systems with data-driven objectives. *arXiv preprint arXiv:1802.06306*, page 11, 2018. (Cited on page 12.)

[74] Q. Liu, L. Li, Z. Tang, and D. Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5356–5366, 2018. (Cited on page 95.)

[75] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. (Cited on pages 1, 4, 37, 61, 79, 86, 103, 104, 151, 154, and 174.)

[76] 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 *Proceedings of the Workshop on Learning to Rank for Information Retrieval*, 2007. (Cited on pages 1, 2, 29, 38, and 39.)

[77] A. Lucic, H. Oosterhuis, H. Haned, and M. de Rijke. Actionable interpretability through optimizable counterfactual explanations for tree ensembles. *arXiv preprint arXiv:1911.12199*, 2019. (Cited on page 12.)

[78] 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 *Proceedings of The Web Conference 2020*, pages 463–473, 2020. (Cited on page 134.)

[79] M. Morik, A. Singh, J. Hong, and T. Joachims. Controlling fairness and bias in dynamic learning-to-rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 429–438, 2020. (Cited on page 178.)

[80] H. Oosterhuis and M. de Rijke. Balancing speed and quality in online learning to rank for information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 277–286, 2017. (Cited on pages 12, 38, 41, 44, 49, 50, 51, 63, and 176.)

[81] 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, 2017. (Cited on pages 11, 15, 40, 42, and 62.)

[82] 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, 2018. (Cited on pages 11, 37, 60, 63, 64, 65, 68, 94, 117, 157, 163, 164, and 173.)

[83] H. Oosterhuis and M. de Rijke. Ranking for relevance and display preferences in complex presentation

layouts. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 845–854, 2018. (Cited on page 12.)

[84] 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. (Cited on pages 11, 59, 60, 94, 109, 117, 157, and 163.)

[85] H. Oosterhuis and M. de Rijke. Taking the counterfactual online: Efficient and unbiased online evaluation for ranking. In *Proceedings of the 2020 International Conference on The Theory of Information Retrieval*. ACM, 2020. (Cited on pages 11, 125, and 164.)

[86] H. Oosterhuis and M. de Rijke. Policy-aware unbiased learning to rank for top-k rankings. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2020. (Cited on pages 11, 77, 105, 114, 128, 131, 136, 152, 153, 155, 156, 157, and 164.)

[87] H. Oosterhuis and M. de Rijke. Robust generalization and safe query-specialization in counterfactual learning to rank. In *Submitted to The World Wide Web Conference*. ACM, 2021. (Cited on pages 11 and 101.)

[88] H. Oosterhuis and M. de Rijke. Unifying online and counterfactual learning to rank. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM'21)*. ACM, 2021. (Cited on pages 11 and 151.)

[89] H. Oosterhuis, S. Ravi, and M. Bendersky. Semantic video trailers. *arXiv preprint arXiv:1609.01819*, 2016. (Cited on page 12.)

[90] H. Oosterhuis, A. Schuth, and M. de Rijke. Probabilistic multileave gradient descent. In *European Conference on Information Retrieval*, pages 661–668. Springer, 2016. (Cited on pages 5, 12, 22, 34, 40, 48, 49, 51, 62, 64, 67, 68, and 173.)

[91] H. Oosterhuis, J. S. Culpepper, and M. de Rijke. The potential of learned index structures for index compression. In *Proceedings of the 23rd Australasian Document Computing Symposium*, pages 1–4, 2018. (Cited on page 12.)

[92] Z. Ovaisi, R. Ahsan, Y. Zhang, K. Vasilaky, and E. Zheleva. Correcting for selection bias in learning-to-rank systems. *arXiv preprint arXiv:2001.11358*, 2020. (Cited on pages 3, 6, 128, 152, 153, and 163.)

[93] A. B. Owen. Monte Carlo theory, methods and examples. *Monte Carlo Theory, Methods and Examples. Art Owen*, 2013. (Cited on page 43.)

[94] E. Politou, E. Alepis, and C. Patsakis. Forgetting personal data and revoking consent under the GDPR: Challenges and proposed solutions. *Journal of Cybersecurity*, 4(1), 2018. (Cited on page 2.)

[95] T. Qin and T.-Y. Liu. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013. (Cited on pages 29, 39, 48, 60, 61, 67, 89, 104, 109, 136, and 152.)

[96] 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, 2013. (Cited on pages 4, 17, 19, 21, 22, 60, 62, 131, 146, 172, 176, and 177.)

[97] F. Radlinski and N. Craswell. A theoretical framework for conversational search. In *Proceedings of the 2017 Conference on Human Information Interaction and Retrieval*, pages 117–126, 2017. (Cited on page 38.)

[98] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning*, pages 784–791, 2008. (Cited on pages 40 and 178.)

[99] 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. (Cited on pages 2, 4, 17, 21, 39, 130, 131, 144, 172, 176, and 177.)

[100] K. Raman, T. Joachims, P. Shivaswamy, and T. Schnabel. Stable coactive learning via perturbation. In *International Conference on Machine Learning*, pages 837–845, 2013. (Cited on pages 2 and 5.)

[101] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997. (Cited on page 1.)

[102] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pages 521–530, 2007. (Cited on page 140.)

[103] A. Roegiest, G. V. Cormack, C. L. Clarke, and M. R. Grossman. TREC 2015 total recall track overview. In *TREC*, 2015. (Cited on page 1.)

[104] M. Sanderson. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375, 2010. (Cited on pages 1, 2, 17, 38, 39, 40, 60, 61, 104,

and 152.)

[105] M. Sanderson, M. L. Paramita, P. Clough, and E. Kanoulas. Do user preferences and evaluation measures line up? In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 555–562, 2010. (Cited on page 1.)

[106] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 158–166, 1999. (Cited on page 1.)

[107] 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 Machine Learning - Volume 48*, pages 1670–1679, 2016. (Cited on page 95.)

[108] A. Schuth, F. Sietsma, S. Whiteson, D. Lefortier, and M. de Rijke. Multileaved comparisons for fast online evaluation. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, pages 71–80, 2014. (Cited on pages 4, 15, 16, 17, 20, 21, 22, 23, 28, 29, 30, 40, 62, 171, 172, and 176.)

[109] A. Schuth, R.-J. Bruintjes, F. Buüttner, J. van Doorn, C. Groenland, H. Oosterhuis, C.-N. Tran, B. Veeling, J. van der Velde, R. Wechsler, et al. Probabilistic multileave for online retrieval evaluation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 955–958, 2015. (Cited on pages 4, 11, 16, 17, 22, 23, 28, 29, 30, 40, 62, and 172.)

[110] A. Schuth, K. Hofmann, and F. Radlinski. Predicting search satisfaction metrics with interleaved comparisons. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 463–472, 2015. (Cited on pages 2, 4, 126, 141, and 179.)

[111] 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, 2016. (Cited on pages 2, 5, 11, 34, 38, 40, 44, 48, 49, 51, 60, 62, 64, 67, 68, 117, 157, 173, 176, and 177.)

[112] I. Shalyminov, O. Dušek, and O. Lemon. Neural response ranking for social conversation: A data-efficient approach. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd Int'l Workshop on Search-Oriented Conversational AI*, pages 1–8, 2018. (Cited on page 78.)

[113] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. In *ACM SIGIR Forum*, volume 33, pages 6–12. ACM New York, NY, USA, 1999. (Cited on page 111.)

[114] A. Slivkins, F. Radlinski, and S. Gollapudi. Ranked bandits in metric spaces: Learning diverse rankings over large document collections. *Journal of Machine Learning Research*, 14(1):399–436, 2013. (Cited on page 40.)

[115] A. Spink, S. Ozmutlu, H. C. Ozmutlu, and B. J. Jansen. US versus European web searching trends. In *ACM Sigir Forum*, volume 36, pages 32–38. ACM New York, NY, USA, 2002. (Cited on page 111.)

[116] A. Swaminathan and T. Joachims. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems*, pages 3231–3239, 2015. (Cited on pages 3 and 92.)

[117] A. Swaminathan, A. Krishnamurthy, A. Agarwal, M. Dudik, J. Langford, D. Jose, and I. Zitouni. Off-policy evaluation for slate recommendation. In *Advances in Neural Information Processing Systems*, pages 3632–3642, 2017. (Cited on page 103.)

[118] B. Szörényi, R. Busa-Fekete, A. Paul, and E. Hüllermeier. Online rank elicitation for plackett-luce: A dueling bandits approach. In *Advances in Neural Information Processing Systems*, pages 604–612, 2015. (Cited on page 42.)

[119] P. S. Thomas, G. Theocharous, and M. Ghavamzadeh. High-confidence off-policy evaluation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. (Cited on pages 102, 106, 107, and 118.)

[120] P. Vakkari and N. Hakala. Changes in relevance criteria and problem stages in task performance. *Journal of Documentation*, 56:540–562, 2000. (Cited on pages 39 and 61.)

[121] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, 2013. (Cited on page 79.)

[122] A. Vardasbi, M. de Rijke, and I. Markov. Cascade model-based propensity estimation for counterfactual learning to rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 2089–2092, 2020. (Cited on page 178.)

[123] A. Vardasbi, H. Oosterhuis, and M. de Rijke. When inverse propensity scoring does not work: Affine corrections for unbiased learning to rank. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2020. (Cited on pages 12, 152, 153, 155, 156, 157, 163,

and 164.)

[124] A. Vlachou, C. Doulkeridis, and K. Nørvåg. Monitoring reverse top-k queries over mobile devices. In *Proceedings of the 10th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 17–24. ACM, 2011. (Cited on page 78.)

[125] H. Wang, R. Langley, S. Kim, E. McCord-Snook, and H. Wang. Efficient exploration of gradient space for online learning to rank. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 145–154. ACM, 2018. (Cited on pages 5, 60, 63, 64, 173, and 176.)

[126] H. Wang, S. Kim, E. McCord-Snook, Q. Wu, and H. Wang. Variance reduction in gradient exploration for online learning to rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 835–844, 2019. (Cited on pages 2, 5, 117, 157, 173, and 176.)

[127] 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, 2016. (Cited on pages 1, 2, 3, 6, 38, 39, 40, 61, 62, 78, 80, 81, 88, 91, 94, 104, 114, 126, 131, 132, 154, 155, 156, and 177.)

[128] 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, 2018. (Cited on pages 78, 81, 88, 94, 131, 134, 152, 153, 156, 172, and 176.)

[129] 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, 2018. (Cited on pages 1, 6, 79, 86, 87, 88, 89, 104, 137, 154, 156, 174, and 177.)

[130] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 159–166, 2007. (Cited on pages 111, 113, and 117.)

[131] Y. Wu, R. Shariff, T. Lattimore, and C. Szepesvári. Conservative bandits. In *International Conference on Machine Learning*, pages 1254–1262, 2016. (Cited on pages 108, 111, 113, and 117.)

[132] 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. (Cited on pages 2, 4, 34, 38, 39, 40, 49, 60, 62, 64, 94, 103, 117, 152, 157, 172, 173, 176, and 177.)

[133] 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, 2010. (Cited on page 29.)

[134] Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th International Conference on World Wide Web*, pages 1011–1018, 2010. (Cited on pages 23, 40, 60, and 61.)

[135] 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, 2016. (Cited on pages 5, 63, 64, and 173.)

[136] S. Zhuang and G. Zuccon. Counterfactual online learning to rank. In *European Conference on Information Retrieval*, pages 415–430. Springer, 2020. (Cited on pages 152, 157, 163, 164, 166, 176, 177, and 178.)

[137] M. Zoghi, S. A. Whiteson, M. De Rijke, and R. Munos. Relative confidence sampling for efficient on-line ranker evaluation. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 73–82, 2014. (Cited on pages 48 and 67.)

[138] 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. (Cited on pages 3, 7, 103, 111, 115, and 174.)

[139] M. Zoghi, T. Tunys, M. Ghavamzadeh, B. Kveton, C. Szepesvari, and Z. Wen. Online learning to rank in stochastic click models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4199–4208, 2017. (Cited on pages 3, 6, 40, and 174.)

# Summary

Ranking systems form the basis for online search engines and recommendation services. They process large collections of items, for instance web pages or e-commerce products, and present the user with a small ordered selection. The goal of a ranking system is to help a user find the items they are looking for with the least amount of effort. Thus the rankings they produce should place the most relevant or preferred items at the top of the ranking. Learning to rank is a field within machine learning that covers methods which optimize ranking systems w.r.t. this goal. Traditional supervised learning to rank methods utilize expert-judgements to evaluate and learn, however, in many situations such judgements are impossible or infeasible to obtain. As a solution, methods have been introduced that perform learning to rank based on user clicks instead. The difficulty with clicks is that they are not only affected by user preferences, but also by what rankings were displayed. Therefore, these methods have to prevent being biased by other factors than user preference. This thesis concerns learning to rank methods based on user clicks and specifically aims to unify the different families of these methods.

The first part of the thesis consists of three chapters that look at online learning to rank algorithms which learn by directly interacting with users. Its first chapter considers large scale evaluation and shows existing methods do not guarantee correctness and user experience, we then introduce a novel method that can guarantee both. The second chapter proposes a novel pairwise method for learning from clicks that contrasts with the previous prevalent dueling-bandit methods. Our experiments show that our pairwise method greatly outperforms the dueling-bandit approach. The third chapter further confirms these findings in an extensive experimental comparison, furthermore, we also show that the theory behind the dueling-bandit approach is unsound w.r.t. deterministic ranking systems.

The second part of the thesis consists of four chapters that look at counterfactual learning to rank algorithms which learn from historically logged click data. Its first chapter takes the existing approach and makes it applicable to top-$k$ settings where not all items can be displayed at once. It also shows that state-of-the-art supervised learning to rank methods can be applied in the counterfactual scenario. The second chapter introduces a method that combines the robust generalization of feature-based models with the high-performance specialization of tabular models. The third chapter looks at evaluation and introduces a method for finding the optimal logging policy that collects click data in a way that minimizes the variance of estimated ranking metrics. By applying this method during the gathering of clicks, one can turn counterfactual evaluation into online evaluation. The fourth chapter proposes a novel counterfactual estimator that considers the possibility that the logging policy has been updated during the gathering of click data. As a result, it can learn much more efficiently when deployed in an online scenario where interventions can take place. The resulting approach is thus both online and counterfactual, our experimental results show that its performance matches the state-of-the-art in both the online and the counterfactual scenario.

As a whole, the second part of this thesis proposes a framework that bridges many gaps between areas of online, counterfactual, and supervised learning to rank. It has taken approaches, previously considered independent, and unified them into a single methodology for widely applicable and effective learning to rank from user clicks.

# Samenvatting

Rankingsystemen vormen de basis voor online zoekmachines en aanbevelingsdiensten. Ze verwerken grote verzamelingen van bijvoorbeeld webpagina's of web-winkel producten, en presenteren een kleine geordende selectie aan de gebruiker. Voor de beste gebruikerservaring, moeten de resulterende rankings de meest relevante of geprefereerde items bovenaan plaatsen. Het learning-to-rank veld omvat methoden die rankingssystemen optimaliseren voor dit doel. Traditionele learning-to-rank methoden maken gebruik van supervisie: annotaties van deskundigen. Omdat het verkrijgen van dergelijke annotaties vaak onmogelijk is, zijn methoden ontwikkeld die leren op basis van gebruikersclicks. Helaas worden clicks niet alleen beïnvloed door gebruikersvoorkeuren, maar ook door welke rankings worden weergegeven. Om werkelijk de gebruikersvoorkeuren te leren moeten deze methoden dus de invloed van zulke andere factoren vermijden. Dit proefschrift betreft learning-to-rank methoden op basis van gebruikersclicks en heeft specifiek het doel de verschillende families van deze methoden te verenigen.

Het eerste deel van dit proefschrift bestaat uit drie hoofdstukken die kijken naar online learning-to-rank algoritmen die leren d.m.v. directe interactie met gebruikers. Het eerste hoofdstuk behandelt evaluatie op grote schaal waar we de eerste methode introduceren die garanties biedt voor zowel juiste resultaten en goede gebruikerservaring. Het tweede hoofdstuk stelt een nieuwe paarsgewijze methode voor om te leren van clicks die in contrast staat met de eerdere dueling-bandits methoden. Onze experimenten tonen aan dat onze paarsgewijze methode veel beter presteert dan de dueling-bandits methoden. Het derde hoofdstuk bevestigt deze bevinding en laat ook zien dat de theorie achter de dueling-bandits methoden incorrect is t.o.v. deterministische rankingsystemen.

Het tweede deel van het proefschrift bestaat uit vier hoofdstukken die kijken naar counterfactual learning-to-rank algoritmen die leren van eerder verzamelde clickdata. Het eerste hoofdstuk maakt deze algoritmen toepasbaar op top-k rankings die niet alle items tegelijkertijd kunnen weergeven. Verder laat het ook zien dat state-of-the-art supervised learning-to-rank methoden toepasbaar zijn in het counterfactual scenario. Het tweede hoofdstuk introduceert een methode die de robuuste generalisatie van 'feature-based' modellen combineert met de hoge prestaties van gespecialiseerde tabelmodellen. Het derde hoofdstuk behandelt evaluatiemethodes en introduceert een methode voor het vinden van de optimale logging-policy die clickdata verzamelt op een manier die de variantie van geschatte rankingmetrieken minimaliseert. Door deze methode toe te passen tijdens het verzamelen van clicks wordt counterfactual evaluatie omgezet in online evaluatie. Het vierde hoofdstuk presenteert een nieuwe counterfactual methode die rekening houdt met de mogelijkheid dat de logging policy niet constant is. Als gevolg hiervan kan het veel efficiënter leren in een online scenario waar interventies plaatsvinden. Onze experimentele resultaten laten zien dat de prestaties van deze methode overeenkomen met de state-of-the-art in zowel het online als het counterfactual scenario.

In zijn geheel stelt het tweede deel van dit proefschrift een raamwerk voor dat veel hiaten overbrugt tussen de online, counterfactual en supervised learning-to-rank gebieden. Het heeft methodes die voorheen als onafhankelijk werden beschouwd, verenigd in één enkele methodologie voor breed toepasbare en effectieve learning-to-rank op basis van gebruikersclicks.