UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Robust Generalization and Safe Query-specialization in Counterfactual Learning to Rank

Oosterhuis, H.; de Rijke, M.

DOI
10.1145/3442381.3450018

Publication date
2021

Document Version
Final published version

Published in
The Web Conference 2021

License
CC BY

Link to publication

Citation for published version (APA):
Oosterhuis, H., & de Rijke, M. (2021). Robust Generalization and Safe Query-specialization in Counterfactual Learning to Rank. In *The Web Conference 2021: proceedings of the World Wide Web Conference WWW 2021 : April 19-23, 2021, Ljubljana, Slovenia* (pp. 158-170). Association for Computing Machinery. https://doi.org/10.1145/3442381.3450018

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:10 Mar 2023

Robust Generalization and Safe Query-Specialization in Counterfactual Learning to Rank

Harrie Oosterhuis
Radboud University
Nijmegen, The Netherlands
harrie.oosterhuis@ru.nl

Maarten de Rijke
University of Amsterdam & Ahold Delhaize
Amsterdam, The Netherlands
derijke@uva.nl

ABSTRACT

Existing work in counterfactual Learning to Rank (LTR) has focussed on optimizing feature-based models that predict the optimal ranking based on document features. LTR methods based on bandit algorithms often optimize tabular models that memorize the optimal ranking per query. These types of model have their own advantages and disadvantages. Feature-based models provide very robust performance across many queries, including those previously unseen, however, the available features often limit the rankings the model can predict. In contrast, tabular models can converge on any possible ranking through memorization. However, memorization is extremely prone to noise, which makes tabular models reliable only when large numbers of user interactions are available. Can we develop a robust counterfactual LTR method that pursues memorization-based optimization whenever it is safe to do?

We introduce the Generalization and Specialization (GENSPEC) algorithm, a robust feature-based counterfactual LTR method that pursues per-query memorization when it is safe to do so. GENSPEC optimizes a single feature-based model for generalization: robust performance across all queries, and many tabular models for specialization: each optimized for high performance on a single query. GENSPEC uses novel relative high-confidence bounds to choose which model to deploy per query. By doing so, GENSPEC enjoys the high performance of successfully specialized tabular models with the robustness of a generalized feature-based model. Our results show that GENSPEC leads to optimal performance on queries with sufficient click data, while having robust behavior on queries with little or noisy data.

ACM Reference Format:
Harrie Oosterhuis and Maarten de Rijke. 2021. Robust Generalization and Safe Query-Specialization in Counterfactual Learning to Rank. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3442381.3450018

1 INTRODUCTION

Ranking systems form the basis for search and recommendation services on the world wide web [30]. The field of Learning to Rank (LTR) considers methods that optimize ranking systems [13, 21, 31]. An important branch of LTR research is based on learning from user interactions [10, 12, 14, 15]. This approach has several advantages. For a service with an active user base, user interactions are virtually free and widely available [15]. Interactions allow methods to closely learn user preferences [30], even in cases where expert annotations cannot be obtained [39]. However, user interactions are affected by noise and bias. Interactions on rankings are particularly affected by *position bias* [7]: items often receive more clicks due to their display-position, and not due to being more preferred by users. Therefore LTR methods that learn from user interactions have to correct for the forms of bias that affect them [15].

In previous work, these LTR methods have been divided into online and counterfactual approaches [4, 11, 12], where online approaches learn from direct interactions [23, 44, 45], and counterfactual approaches learn from historical interaction data [15, 25, 39]. While this division is very interesting [4, 12, 26], this paper focusses on a different division between methods that learn feature-based models and those that learn tabular models. The former group of methods optimize models that predict the optimal ranking based on the available document features, this group includes most work on counterfactual LTR [15, 25, 39] and online LTR methods such as Dueling Bandit Gradient Descent [44], Pairwise Differentiable Gradient Descent [23] and the Counterfactual Online Learning to Rank algorithm [45]. The latter group of methods do not predict based on document features, instead they attempt to memorize the optimal ranking. Thus they try to optimize the ranking directly, this group includes methods such as Position-Based Model algorithm (PBM) [18], the Hotfix algorithm [46], and various approaches based on k-armed bandit algorithms [16, 17, 19, 20, 28].

Feature-based models are very good at *generalization*, they perform well across large groups of queries including rare or previously unseen ones [21]. However, feature-based models usually do not reach optimal performance because they are limited by the available features [46]. For instance, often the features do not contain enough information to predict the optimal ranking. In contrast, tabular models are well-suited for *specialization*, they can reach optimal performance on individual queries where enough interactions are available [18]. Because they do not utilize feature-based predictions, tabular models can converge on any possible ranking, and thus provide extremely high performance if enough interaction data is available [46]. Unfortunately, tabular models can also provide extremely poor performance when not enough data has been gathered. This happens because the ranking behavior memorized on one query cannot be generalized to other queries [46]. For instance, tabular models have no preference between rankings on previously unseen queries. Moreover, because their behavior does not generalize, tabular models are very sensitive to noise on infrequent queries where little data is available.

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW '21, April 19–23, 2021, Ljubljana, Slovenia
© 2021 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450018

Therefore, the important choice between feature-based models and tabular models should mainly depend on the amount of available data per query. Feature-based models are the best choice when little or no interactions on a query are available, while tabular models are the better choice on queries where large numbers of interactions have been gathered. This observation has been made in previous work, i.e., Zoghi et al. [46] noted that their Hotfix algorithm should only be applied to underperforming *torso-queries*, and explicitly advised not to apply the algorithm to infrequent *tail* queries. In practice, tabular models are applied to queries for which feature-based ranking systems appear to be underperforming and that receive enough interactions so that improvements can be found [9, 22, 36]. To the best of our knowledge, there is no theory-grounded approach for deciding when it is safe to pursue optimizations offered by a tabular model over the robustness offered by a feature-based model, despite the fact that these choices are being made on a daily basis in real-world production systems.

Inspired by the advantages of both types of models and the lack of a principled approach to choosing and switching between them, we introduce the *Generalization and Specialization* (GENSPEC) framework. Using counterfactual LTR, GENSPEC optimizes multiple ranking models for either generalization across queries or specialization in a specific query. Using the available click data GENSPEC simultaneously trains: (i) a generalized feature-based model that performs well across all queries, and (ii) a specialized tabular model that memorizes a ranking for each observed query. Per individual query there is the choice between three models: (i) the logging policy model used to gather the click data, (ii) the generalized feature-based model, and (iii) the specialized tabular model. To reliably choose between these models, we introduce novel high-confidence bounds on the relative performance between models, based on existing bounds on absolute performance [35]. Using these novel bounds, GENSPEC chooses conservatively: the feature-based model is only deployed when there is high confidence that it outperforms the logging policy; and a tabular model overrules the other models on a single query, if with high confidence it is expected to outperform both for that specific query. By simultaneously deploying both feature-based and tabular models, GENSPEC exploits the advantageous properties of both: the robustness of a generalized feature-based model with the potential high performance of a specialized tabular model.

The main contributions of this work are:

- the GENSPEC framework that simultaneously optimizes generalized and specialized models and decides which to deploy per individual query; and
- a novel high-confidence bound for estimating the relative performance difference between ranking models.

To the best of our knowledge, GENSPEC is the first framework to simultaneously optimize both feature-based and tabular models based on user interactions, and reliably choose between them on a query-level using high-confidence bounds.

2 RELATED WORK

LTR or the optimization of ranking models w.r.t. ranking metrics, is a well-established field within Information Retrieval (IR) [21]. Supervised LTR methods make use of annotated datasets based on expert judgements. These datasets contain labels indicating the expert-judged level of relevance for numerous query-document pairs [6, 8, 27]. With such an annotated dataset evaluation and optimization is mostly straightforward. Since ranking metrics are not differentiable, LTR methods either optimize an approximate metric or a lower bound on the metric [1, 21, 41].

While supervised LTR has a very important place in the IR field, several limitations of the supervised approach have become apparent. Most importantly, expert annotations often disagree with actual user preferences [30]. Furthermore, they are expensive and time-consuming to gather [6, 27]. In privacy-sensitive settings it is often impossible to acquire judgements without breaching the privacy of users, i.e., in search through emails or personal documents [39]. Consequently, interest in LTR from user interactions has increased rapidly in recent years [2, 15, 25, 37, 39]. In contrast to annotated datasets, interactions are cheap and easy to obtain if a service has active users, and can be gathered without showing privacy-sensitive content to experts [15, 39]. Moreover, interactions are indicative of actual user preferences [29]. Unfortunately, unlike annotations, interactions are noisy and biased indicators of relevance and thus bring their own difficulties [7].

The idea of LTR from user clicks goes back to one of the earliest pairwise LTR methods [14]. More recent work has introduced the field of counterfactual LTR [15, 39], where Inverse Propensity Scoring (IPS) is used to counter the effect of position bias [7]. The underlying idea is that a model of position bias can be inferred from user interactions reliably, and with such a model IPS can correct for its effect [40] (see Section 3). Work on counterfactual LTR is concentrated around methods for estimating models of bias [3, 40], and counterfactual estimators that use these models for unbiased evaluation and optimization [1, 15, 39]. Recent work has extended the counterfactual LTR approach to also correct for item-selection bias [25] and trust bias [2, 37]. Existing work on counterfactual LTR has only optimized feature-based models, e.g., support vector machines [15], linear ranking models [25], and neural networks [1].

In contrast, methods that optimize tabular models are common in online LTR work [16, 17, 19, 20, 28]. These methods learn from direct interactions with the user where they can decide which rankings will be displayed. Zoghi et al. [46] have argued that the main advantage of tabular models is that they are not limited by the available features. Consequently, they can converge on any possible ranking, allowing for extremely high performance if successfully optimized. However, tabular models do not generalize across queries, and as a result, these methods see each query as an independent ranking problem. For this reason, these algorithms have a cold-start problem: they often have an initial period of very poor performance. To mitigate this, various heuristics (e.g., in terms of business rules, query frequency, or sampling strategies for training) are put in place when using tabular models in practice [9, 33, 46].

Prior work on query frequency has found that they follow a long-tail distribution [32, 34]. White et al. [42] found that 97% of queries received fewer than 10 clicks over six months. For such infrequent queries, the performance of a tabular model may never leave the initial period of poor performance [43]. Even though it is known that deployment should be avoided in such cases, to the best of our knowledge, there exist no theoretically principled method for detecting when it is safe to deploy a tabular model. The only existing method that safely chooses between models appears to be the Safe Exploration Algorithm (SEA) [11], which applies high-confidence bounds to the performance of a safe logging policy model and a newly learned ranking model. If these bounds do not overlap, SEA can conclude with high-confidence that one model outperforms the other. So far, no work has used SEA for the deployment of tabular models in LTR.

3 BACKGROUND

3.1 The Learning to Rank Task

The goal of LTR is to optimize a ranking model w.r.t. to a ranking metric. We use y to denote a ranking, which is simply an ordering of documents d: $y = [d_1, d_2, \ldots]$. For a ranking model, we use π and $\pi(y \mid q)$ for the probability of π displaying y for query q:

$$\pi(y \mid q) = P(Y = y \mid q, \pi). \tag{1}$$

Queries follow a distribution determined by the users, we use $P(Q = q)$ to denote the probability that a user-issued query is q. Furthermore, we use $r(q, d)$ to denote the relevance of d w.r.t. q, this is the probability that a user considers d a relevant result for query q: $r(q, d) = P(R = 1 \mid q, d)$. Most ranking metrics compute the quality of a single ranking as a weighted sum over the ranks [15, 21]. Let $rank(d \mid y) \in \mathbb{Z}_{>0}$ indicate the rank of d in y and $\lambda : \mathbb{Z}_{>0} \rightarrow \mathbb{R}$ be an arbitrary weight function, then we use $\Delta(y \mid q, r)$ to denote the quality of y w.r.t. the query q and the relevance function r:

$$\Delta(y \mid q, r) = \sum_{d \in y} \lambda\big(rank(d \mid y)\big) \cdot r(q, d). \tag{2}$$

Ranking metrics differ in their choice of λ and thus they differ in how important they consider each rank to be. A common ranking metric is the *Discounted Cumulative Gain* (DCG) metric [13]; λ can be chosen accordingly:

$$\lambda^{DCG}\big(rank(d \mid y)\big) = \log_2\big(1 + rank(d \mid y)\big)^{-1}. \tag{3}$$

Finally, the performance of a ranking model π according to a ranking metric is an expectation over the query distribution $P(Q = q)$ and the model behavior, that is, the expected ranking quality $\Delta(y \mid q, r)$ w.r.t. the model π and the query distribution:

$$\mathcal{R}(\pi) = \int \Big(\sum_{y \in \pi} \Delta(y \mid q, r) \cdot \pi(y \mid q) \Big) P(Q = q) \, dq. \tag{4}$$

Supervised LTR methods use datasets where often $P(Q = q)$ is approximated based on logged user-issued queries, and the relevance function $r(q, d)$ is estimated using expert judgements. Given such a dataset, these methods can optimize the resulting estimate of $\mathcal{R}(\pi)$ in a supervised manner [5, 21, 41].

3.2 Counterfactual Learning to Rank

As discussed in Section 2, there are severe limitations to estimates of the relevance function $r(q, d)$ based on expert judgements. Acquiring relevance annotations is expensive, time-consuming [6, 27], and sometimes infeasible, for instance, due to privacy concerns [39]. Moreover, the resulting annotations are often not aligned with the actual user preferences [30].

An attractive alternative to the approach of collecting expert generated annotations is counterfactual LTR, which learns from historical interaction logs [15, 39]. Let π_0 be the logging policy that was used for gathering interactions, with q_i as the user-issued query, y_i as the ranking displayed at interaction i and: $y_i \sim \pi_0(y \mid q_i)$. To model position bias, we will use the $o_i(d) \in \{0, 1\}$ to indicate whether item d was examined by the user or not: $o_i(d) \sim P\big(O \mid d, y_i\big)$. Furthermore, we use $c_i(d) \in \{0, 1\}$ to indicate whether d was clicked at time step i: $c_i(d) \sim P\big(C \mid o_i(d), r(d \mid q)\big)$. We follow the common assumption that users do not click on unexamined documents:

$$P\big(C = 1 \mid o_i(d) = 0, r(q, d)\big) = 0. \tag{5}$$

Given that a document is observed, its click probability is assumed to be proportional to its relevance with some constant offset $\mu \in \mathbb{R}_{>0}$:

$$P\big(C = 1 \mid o_i(d) = 1, r(q, d)\big) \propto r(q, d) + \mu. \tag{6}$$

The data used in counterfactual LTR consists of the observed clicks c, the displayed rankings y, the propensity scores ρ, and the issued queries q for N interactions:

$$\mathcal{D} = \big\{(c_i, y_i, \rho_i, q_i)\big\}_{i=1}^{N}. \tag{7}$$

We apply the policy-aware approach [25] and base the propensity scores both on the logging policy π_0 as the position bias of the user:

$$\rho_i(d) = \sum_{y \in \pi_0} P\big(O = 1 \mid d, y_i\big) \cdot \pi_0(y_i \mid q_i). \tag{8}$$

The main difficulty in learning from $\mathcal{D}$ is that the observed clicks c are affected by both relevance, the logging policy behavior, and the users' position bias. Counterfactual LTR methods apply IPS estimators to correct the click signal for the position bias and logging policy behavior. The resulting estimated reward can then be used to unbiasedly optimize a ranking model.

The estimated reward is an average over an IPS transformation of each point of interaction data:

$$\hat{\mathcal{R}}(\pi \mid \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \sum_{y \in \pi} \hat{\Delta}(y \mid c_i, \rho_i) \cdot \pi(y \mid q_i), \tag{9}$$

where $\hat{\Delta}$ is an IPS estimator:

$$\hat{\Delta}(y \mid c_i, \rho_i) = \sum_{d \in y} \lambda\big(rank(d \mid y)\big) \cdot \frac{c_i(d)}{\rho_i(d)}. \tag{10}$$

The estimated reward $\hat{\mathcal{R}}(\pi \mid \mathcal{D})$ is proven to enable unbiased LTR because it maintains the ordering of ranking models (see the proof in Appendix A). In other words, for any two ranking models π_1, π_2 the estimated reward $\hat{\mathcal{R}}$ will prefer the same model as the true reward $\mathcal{R}$:

$$\forall \pi_1, \pi_2,\ \mathbb{E}\big[\hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) > \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D})\big] \leftrightarrow \mathcal{R}(\pi_1) > \mathcal{R}(\pi_2). \tag{11}$$

This implies that in expectation both share the same optima:

$$\arg\max_{\pi} \mathbb{E}\big[\hat{\mathcal{R}}(\pi \mid \mathcal{D})\big] = \arg\max_{\pi} \mathcal{R}(\pi). \tag{12}$$

Therefore, optimizing a ranking model π w.r.t. $\hat{\mathcal{R}}$ is expected to optimize it w.r.t. $\mathcal{R}$ as well, thus allowing for unbiased optimization. Previous work has introduced several methods for maximizing $\hat{\mathcal{R}}$ so as to optimize different LTR metrics [1, 15, 25].

4 METHOD

This section introduces our *Generalization and Specialization* (GENSPEC) framework for query-specialization in counterfactual LTR. First, we describe how a feature-based model is optimized for generalization across queries. Second, we explain how we optimize tabular models for specialized performance on individual queries. Third, we introduce novel high-confidence bounds on relative performance, allowing us to reliably estimate performance differences between models. Fourth, we propose the GENSPEC meta-policy that chooses between the optimized models using the novel high-confidence bounds. Finally, the section concludes with a summary of the GENSPEC framework.

4.1 Feature-Based Query-Generalization

Feature-based LTR optimizes models that try to predict the optimal model from query-document features. There are many feature-based models one can choose; in LTR linear models, support-vector-machines, neural networks and decision tree ensembles are popular choices. All of these models can be conceptualized as a function $f_\theta(q,d) \in \mathbb{R}$ that transforms the features available for query q and document d into a score based on the learned parameters θ. Rankings are then constructed by sorting all documents d for a query q according to their predicted scores.

In practice, a feature-based ranking model often does not sort every document in the collection [see, e.g., 38]; instead, earlier steps pre-select only a subset of documents that make it into the final ranking. We use $(d,d') \in q$ as shorthand to denote that both documents d and d' have made it through the pre-selection. Furthermore, we use $d \prec_y d'$ to indicate that d is ranked higher than d in ranking y. To deal with ties between documents in their predicted scores, we introduce the set of valid rankings according to f_θ:

$$Y_\theta(q) = \Big\{y \mid \forall (d,d') \in q, \big(f_\theta(q,d) > f_\theta(q,d') \rightarrow d \prec_y d'\big)\Big\}. \quad (13)$$

The final feature-based ranking model simply chooses uniform randomly from the set of valid rankings:

$$\pi_\theta(y \mid q) = \begin{cases} \frac{1}{|Y_\theta(q)|} & \text{if } y \in Y_\theta(q), \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

In theory, the optimal set of parameters maximizes the true reward:

$$\theta^* = \arg\max_\theta R(\pi_\theta). \quad (15)$$

In practice, counterfactual LTR optimizes θ w.r.t. the estimated reward: $\hat{R}(\pi_\theta \mid \mathcal{D})$. Many LTR methods can be applied, we apply stochastic gradient descent on the counterfactual version of LambdaLoss [41] as described by Oosterhuis and de Rijke [25]. Importantly, to avoid overfitting $\mathcal{D}$ is divided into a training and evaluation partition, a standard practice in machine learning. The parameters θ are found using gradients based on the training partition, but chosen to maximize the expected reward on the validation partition. As a result, θ is expected to maximize the reward on previously unseen queries. This choice leads to a ranking model π_θ that has robust behavior that generalizes across queries. Feature-based models are particularly suited for query-generalization because they can apply behavior learned on one query to any other query, including those previously unseen.

4.2 Tabular Query-Specialization

While feature-based models are good at generalization, they are often limited by the available features. In many cases, the available features do not provide enough information to predict the optimal ranking. This limitation does not apply to tabular models, which try to memorize the optimal rankings instead of predicting them from features [46]. Because they do not depend on features, tabular models can cover all possible permutations of documents. However, memorization does not generalize, i.e., the ranking behavior learned on one query cannot be applied to another. Instead, tabular models are well suited for query-specialization: learning extremely high-performing behavior for individual queries.

As discussed in Section 2, many online LTR algorithms optimize tabular models for the rankings of individual queries. We will counterfactually estimate the relevance of every document based on the available data $\mathcal{D}$:

$$\hat{r}(q,d \mid \mathcal{D}) = \frac{1}{\sum_{i \in \mathcal{D}} \mathbb{1}[q_i = q]} \sum_{i \in \mathcal{D}} \mathbb{1}[q_i = q] \cdot \frac{c_i(d)}{\rho_i(d)}, \quad (16)$$

resulting in a tabular model that stores $\hat{r}(q,d \mid \mathcal{D})$ for every query-document pair observed in $\mathcal{D}$. We note that this counterfactual estimate is also used by the PBM [18], although PBM applies it to data gathered by its bandit algorithm. Again, we use a set of valid rankings to deal with cases where multiple documents have the same estimated relevance:

$$Y_\mathcal{D}(q) = \Big\{y \mid \forall (d,d') \in q, \big(\hat{r}(q,d \mid \mathcal{D}) > \hat{r}(q,d' \mid \mathcal{D}) \rightarrow d \prec_y d'\big)\Big\}.$$

The resulting ranking model simply chooses uniform randomly from the set of valid rankings:

$$\pi_\mathcal{D}(y \mid q) = \begin{cases} \frac{1}{|Y_\mathcal{D}(q)|} & \text{if } y \in Y_\mathcal{D}(q), \\ 0 & \text{otherwise.} \end{cases} \quad (17)$$

Importantly, $\pi_\mathcal{D}$ always maximizes the estimated reward:

$$\hat{R}(\pi_\mathcal{D} \mid \mathcal{D}) = \max_\pi \hat{R}(\pi \mid \mathcal{D}). \quad (18)$$

This is both a benefit and a risk: on the one hand, it means that $\pi_\mathcal{D}$ is optimal if the estimate is correct, i.e., $\mathcal{D}$ is large enough for an accurate $\hat{R}(\pi_\mathcal{D} \mid \mathcal{D})$; on the other hand, $\pi_\mathcal{D}$ is extremely sensitive to noise since it is completely overfitting on $\mathcal{D}$. Therefore, if $\mathcal{D}$ is small and the estimate is probably highly incorrect due to noise, $\pi_\mathcal{D}$ is likely to have very poor performance. For instance, in the extreme case that only a single click is available in $\mathcal{D}$ for a query, $\pi_\mathcal{D}$ will place the clicked document on top of the ranking for that query, despite a single click being very poor evidence of a document being relevant. Moreover, if no clicks are available for a query, then $\pi_\mathcal{D}$ will uniform randomly select any possible permutation.

In stark contrast with the feature-based π_θ, the clicks related to one query will never affect the behavior of $\pi_\mathcal{D}$ w.r.t. any other query. Thus, one could also think of $\pi_\mathcal{D}$ as an ensemble of models each specialized in a single query. This is very similar to the behavior of online LTR algorithms that optimize tabular models, as they often see each query as an individual ranking problem. Similarly, $\pi_\mathcal{D}$ can have extremely high performance for one query, while also displaying detrimental behavior for another.

4.3 Reliably Choosing Between Models

So far, we have introduced the feature-based ranking model π_θ and the tabular ranking model $\pi_\mathcal{D}$. It also appears that there is no clear optimal choice between π_θ and $\pi_\mathcal{D}$; instead, this choice seems to mostly depend on the available data $\mathcal{D}$. We wish to deploy the model that leads to the highest performance, however, we also want to avoid a detrimental user experience due to choosing the wrong model. Recent work by Jagerman et al. [11] introduced the Safe Exploration Algorithm (SEA), for choosing safely between a safe model and a risky learned model. SEA applies high confidence bounds [35] to the performances of both models. The risky learned model is only deployed if the lower bound on its performance is greater than the upper bound of the safe initial model. In other words, if with high confidence SEA can conclude that deploying the risky model does not lead to a decrease in performance.

We want to apply a similar strategy to choose between π_θ and $\pi_\mathcal{D}$, however, we note that SEA uses two bounds to make a single decision. This is not necessary and instead we introduce a novel bound on the relative performance difference between two models. By using only a single bound, the high-confidence decision can be made with considerably less data.

Let π_1 and π_2 be any two models, and let $\delta(\pi_1, \pi_2)$ indicate the true difference in performance between them:

$$\delta(\pi_1, \pi_2) = \mathcal{R}(\pi_1) - \mathcal{R}(\pi_2). \tag{19}$$

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

$$\hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) = \hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}). \tag{20}$$

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

$$R_{i,d} = \frac{c_i(d)}{\rho_i(d)} \sum_{y \in \pi_1 \cup \pi_2} \big(\pi_1(y \mid q_i) - \pi_2(y \mid q_i)\big) \cdot \lambda\big(rank(d \mid y)\big). \tag{21}$$

Then, for computational efficiency we rewrite:

$$\hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \sum_{d \in y_i} R_{i,d} = \frac{1}{|\mathcal{D}|K} \sum_{i \in \mathcal{D}} \sum_{d \in y_i} K \cdot R_{i,d}. \tag{22}$$

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

$$v = \frac{2|\mathcal{D}|K \ln\left(\frac{2}{1-\epsilon}\right)}{|\mathcal{D}|K - 1} \sum_{i \in \mathcal{D}} \sum_{d \in y_i} \left(K \cdot R_{i,d} - \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D})\right)^2, \tag{23}$$

we follow Thomas et al. [35] to get the high-confidence bound:

$$CB(\pi_1, \pi_2 \mid \mathcal{D}) = \frac{7Kb \ln\left(\frac{2}{1-\epsilon}\right)}{3(|\mathcal{D}|K - 1)} + \frac{1}{|\mathcal{D}|K} \cdot \sqrt{v}. \tag{24}$$

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

$$\begin{aligned} LCB(\pi_1, \pi_2 \mid \mathcal{D}) &= \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) - CB(\pi_1, \pi_2 \mid \mathcal{D}) \\ UCB(\pi_1, \pi_2 \mid \mathcal{D}) &= \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) + CB(\pi_1, \pi_2 \mid \mathcal{D}). \end{aligned} \tag{25}$$

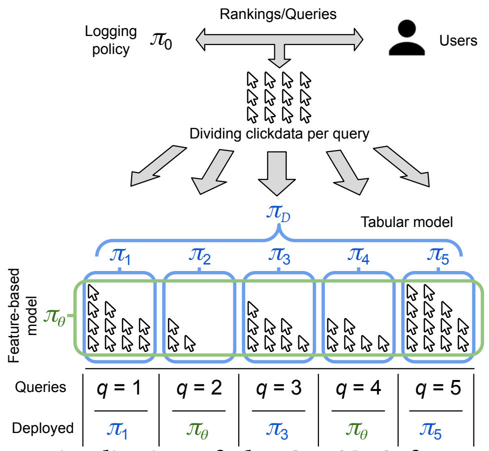

Figure 1: Visualization of the GENSPEC framework. A feature-based model π_θ is trained on the complete dataset, the tabular model $\pi_\mathcal{D}$ consists of many specialized models $\pi_1, \pi_2, \ldots$ each highly-specialized for a single query. GENSPEC decides which model to deploy per query, based on high-confidence bounds.

As proven by Thomas et al. [35], with at least a probability of ϵ they bound the true value of $\delta(\pi_1, \pi_2)$:

$$P\Big(\delta(\pi_1, \pi_2) \in \big[LCB(\pi_1, \pi_2 \mid \mathcal{D}), UCB(\pi_1, \pi_2 \mid \mathcal{D})\big]\Big) > \epsilon. \tag{26}$$

In other words, if $LCB(\pi_1, \pi_2 \mid \mathcal{D}) > 0$, the probability that π_1 outperforms π_2 is higher than ϵ, i.e., $P(\mathcal{R}(\pi_1) > \mathcal{R}(\pi_2)) > \epsilon$.

We note that the novelty of our bound is that it bounds the *performance difference of two models*, instead of the *performance of individual models* used in earlier work by Jagerman et al. [11] and Thomas et al. [35]. Whereas the SEA approach uses two bounds, our approach can decide between models with only a single bound. In Appendix B we theoretically analyze the difference between these approaches and conclude that our relative bound is more data-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. In practice, this means that our novel bound requires much less data to identify a better performing model than SEA.

4.4 Generalization and Specialization

So far we have described the four main ingredients of the GENSPEC Framework: (i) the logging policy π_0 used to gather the click data $\mathcal{D}$, π_0 is assumed to be safe to deploy w.r.t. user experience; (ii) the feature-based ranking model π_θ optimized for robust performance generalized across queries; (iii) the tabular ranking model $\pi_\mathcal{D}$ that specializes by memorizing a ranking per query; and (iv) the novel high-confidence bound on model performance differences that can be used to choose between models. GENSPEC will choose between deploying π_0, π_θ, and $\pi_\mathcal{D}$ on a per query basis by applying a doubly conservative strategy: π_θ is only deployed when there is high-confidence that it outperforms π_0 across all queries; $\pi_\mathcal{D}$ is only deployed for a query q when there is high-confidence that it outperforms π_0 and π_θ on the same query q.

However, to avoid overfitting we should not use the same data $\mathcal{D}$ for both the training of models and to compute performance bounds.

This is especially important because $\pi_{\mathcal{D}}$ will completely overfit on $\mathcal{D}$, thus if evaluated on the same $\mathcal{D}$ the performance of $\pi_{\mathcal{D}}$ will always appear optimal (Eq. 18). To avoid this overfitting problem, we split $\mathcal{D}$ in a training partition $\mathcal{D}^{train}$ and a model-selection partition $\mathcal{D}^{sel}$ so that $\mathcal{D} = \mathcal{D}^{train} \cup \mathcal{D}^{sel}$ and $\mathcal{D}^{train} \cap \mathcal{D}^{sel} = \emptyset$. For each model, we train two versions: one trained on the entire dataset $\mathcal{D}$, and another only trained on $\mathcal{D}^{train}$. Let π'_θ and $\pi'_{\mathcal{D}}$ indicate the versions of feature-based model and the tabular model trained on $\mathcal{D}^{train}$. These models can now safely be compared on the $\mathcal{D}^{sel}$ partition without risk of overfitting.

The GENSPEC strategy assumes that the performance of π_θ is always greater than π'_θ: $\mathcal{R}(\pi_\theta) > \mathcal{R}(\pi'_\theta)$, and similarly: $\mathcal{R}(\pi_{\mathcal{D}}) > \mathcal{R}(\pi'_{\mathcal{D}})$. This is a reasonable assumption since π_θ is trained on a superset of the data on which π'_θ is trained. Using this assumption, GENSPEC chooses between the feature-based model π_θ and the logging policy π_0, using bounds computed on π'_θ:

$$\pi_G(y \mid q) = \begin{cases} \pi_\theta(y \mid q), & \text{if } LCB(\pi'_\theta, \pi_0 \mid \mathcal{D}^{sel}) > 0, \\ \pi_0(y \mid q), & \text{otherwise.} \end{cases} \tag{27}$$

Therefore, $\pi_G = \pi_\theta$ only if with high confidence π'_θ outperforms π_0, otherwise $\pi_G = \pi_0$, i.e., the safe option is chosen. By using π'_θ to decide whether to choose π_θ over π_0, we avoid the overfitting problem while also utilizing the expected higher performance of π_θ over π'_θ.

The next step is to decide between deploying $\pi_{\mathcal{D}}$ and π_G. Again to avoid overfitting, we use π'_G for a copy of π_G that is only based on $\mathcal{D}^{train}$:

$$\pi'_G(y \mid q) = \begin{cases} \pi'_\theta(y \mid q), & \text{if } LCB(\pi'_\theta, \pi_0 \mid \mathcal{D}^{sel}) > 0, \\ \pi_0(y \mid q), & \text{otherwise.} \end{cases} \tag{28}$$

Remember that the behavior of tabular ranking models like $\pi_{\mathcal{D}}$ is independent per query, i.e., clicks related to one query will never affect the ranking behavior of $\pi_{\mathcal{D}}$ w.r.t. any other query. For this reason, the choice between $\pi_{\mathcal{D}}$ and π_G is made on a per query basis, unlike the choice between the generalized π_θ and safe π_0. To do so, we divide $\mathcal{D}^{sel}$ per query, resulting in a $\mathcal{D}^{sel}_q$ per query q:

$$\mathcal{D}^{sel}_q = \{(c_i, y_i, \rho_i, q_i) \in \mathcal{D}^{sel} \mid q_i = q\}. \tag{29}$$

This allows us to bound the relative performance between $\pi'_{\mathcal{D}}$ and π'_G w.r.t. a single query q. Finally, we can use this to choose between deploying the highly-specialized $\pi_{\mathcal{D}}$ and the robust generalized π_G per query:

$$\pi_{GS}(y \mid q) = \begin{cases} \pi_{\mathcal{D}}(y \mid q), & \text{if } LCB(\pi'_{\mathcal{D}}, \pi'_G \mid \mathcal{D}^{sel}_q) > 0, \\ \pi_G(y \mid q), & \text{otherwise.} \end{cases} \tag{30}$$

Thus $\pi_{GS}(y \mid q) = \pi_{\mathcal{D}}(y \mid q)$ only if there is high-confidence that $\pi'_{\mathcal{D}}$ outperforms π_G. Because this decision is made independently per query, it is entirely possible that for one query π_{GS} deploys $\pi_{\mathcal{D}}$ while deploying π_G for another. This behavior allows π_{GS} to have the high performance of a successfully specialized $\pi_{\mathcal{D}}$ on queries where it is highly confident of such high performance, while also relying on the robust behavior of π_θ where this confidence has not been obtained. We note that because decisions are only made with confidence, there will be some delay between the moment that $\pi_{\mathcal{D}}$ outperforms π_G and when $\pi_{\mathcal{D}}$ is deployed. Nevertheless, GENSPEC safely combines the strongest advantages of each model: the safe behavior of π_0, the robust generalization of feature-based π_θ and high-performance at convergence of the tabular model $\pi_{\mathcal{D}}$.

Algorithm 1 The GENSPEC training and serving procedures.

```
1: procedure INITIALIZE($\pi_0, \mathcal{D}, \epsilon, \beta$)
2:     $\mathcal{D}^{train}, \mathcal{D}^{sel} \leftarrow$ random_split($\mathcal{D}, \beta$)
3:     $\pi'_\theta \leftarrow$ train_feature_based_model($\mathcal{D}^{train}$)
4:     $\pi'_{\mathcal{D}} \leftarrow$ infer_tabular_model($\mathcal{D}^{train}$)
5:     feat_model_activated $\leftarrow$ LCB($\pi'_\theta, \pi_0 \mid \mathcal{D}^{sel}$) $> 0$
6:     override_queries $\leftarrow \{\}$
7:     for $q \in \mathcal{D}$ do                ▷ Loop over unique queries in $\mathcal{D}$.
8:         if feat_model_activated then
9:             if LCB($\pi'_{\mathcal{D}}, \pi'_\theta \mid \mathcal{D}^{sel}_q$) $> 0$ then
10:                override_queries $\leftarrow$ override_queries $\cup \{q\}$
11:        else if LCB($\pi'_{\mathcal{D}}, \pi_0 \mid \mathcal{D}^{sel}_q$) $> 0$ then
12:            override_queries $\leftarrow$ override_queries $\cup \{q\}$
13:    $\pi_\theta \leftarrow$ train_feature_based_model($\mathcal{D}$)
14:    $\pi_{\mathcal{D}} \leftarrow$ infer_tabular_model($\mathcal{D}$)
15: procedure MODEL_TO_SERVE($q$)            ▷ User-issued query $q$.
16:    if $q \in$ override_queries then
17:        return $\pi_{\mathcal{D}}$
18:    else if feat_model_activated then
19:        return $\pi_\theta$
20:    else
21:        return $\pi_0$
```

4.5 Summary

This completes our introduction of the GENSPEC framework, Algorithm 1 summarizes it in pseudocode and Figure 1 provides a visualization. The implementation in Algorithm 1 is divided in two procedures: intialization and serving. The initialization phase trains the models and decides where they will be deployed, subsequently, the serving procedure shows how a model is selected for any incoming user-issued query.

The initialization procedure takes as input: (i) the safe logging policy π_0, (ii) the clickdata $\mathcal{D}$, (iii) the confidence parameter ϵ, and (iv) β the percentage of data to be held-out for $\mathcal{D}^{sel}$ (Line 1). First, $\mathcal{D}$ is divided into training data $\mathcal{D}^{train}$ and selection data $\mathcal{D}^{sel}$ (Line 2), then the feature-based model π'_θ and the tabular model $\pi'_{\mathcal{D}}$ are trained on $\mathcal{D}^{train}$ (Line 3 and 4). The feature-based model is activated if the lower confidence bound on the performance difference between π'_θ and π_0 is positive (Line 5). Next, an empty set is created to keep track of the queries for which the tabular model will override the other models (Line 6). For each unique query q in $\mathcal{D}$, the lower confidence bound on the performance difference for q between $\pi'_{\mathcal{D}}$ and the activated model is computed (Line 9 or 11). If this bound is positive, the query is added to the override set (Line 10 or 12). Finally, another feature-based model π_θ and tabular model $\pi_{\mathcal{D}}$ are trained, this time on all available data $\mathcal{D}$ (Line 13 and 14).

The serving procedure takes as input an incoming user-issued query q. If q is in the override set (Line 16) then $\pi_{\mathcal{D}}$ is used to generate the ranking; if q is not in the override set but the feature-based model is activated then π_θ is used (Line 18); otherwise the safe option π_0 is used (Line 20).

By choosing between the deployment of two types of models: one for generalization and another for specialization, GENSPEC safely combines the high performance at convergence of specialization and the robust safe performance of generalization. In this paper, we discuss how GENSPEC can be used for query-specialization, however, other choices for models and the specialization task can be made. For example, one could choose to optimize many models specialized in the preferences of individual users, and a single feature-based model for generalization across all users. By choosing between models on a per user basis, GENSPEC could be used for safe personalization. Due to this flexibility, we refer to GENSPEC as a *framework* that can be used for a wide variety of safe specialization scenarios.

5 EXPERIMENTAL SETUP

In our experiments we compare GENSPEC to: (i) a single feature-based model and a single tabular model to evaluate if GENSPEC truly safely combines the advantages of generalization and specialization; (ii) the GENSPEC strategy using the SEA bounds, in order to test whether our novel bound on relative performance is truly more efficient; and lastly, (iii) online bandit-style LTR algorithms, to evaluate whether GENSPEC provides the same high-performance convergence while avoiding initial periods of poor performance. In order to perform reproducible experiments under varying circumstances, we make use of a semi-synthetic setup and evaluate both on previously seen and unseen queries.

5.1 The Semi-Synthetic Setup

Our experimental setup is semi-synthetic: 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 and online LTR literature [1, 15, 24]. We make use of the three largest LTR industry datasets: *Yahoo! Webscope* [6], *MSLR-WEB30k* [27], and *Istella* [8]. 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). 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 [15]. Then, user examination is simulated with probabilities inverse to the displayed rank of a document: $P\big(O = 1 \mid d, y\big) = \frac{1}{rank(d \mid y)}$. Finally, user clicks are generated according to the following formula using a single parameter $\alpha \in \mathbb{R}$:

$$P\big(C = 1 \mid o_i(d) = 1, r(q, d)\big) = 0.2 + \alpha \cdot \text{relevance\_label}(q, d). \quad (31)$$

In our experiments, we use $\alpha = 0.2$ and $\alpha = 0.025$; the former represents an easier setting where relevance has a great effect on the click probability; the latter represents a more noisy and harder setting where relevance has a far smaller influence. Clicks are only generated on the training and validation partitions; 50% of the training clicks are separated for model selection ($\mathcal{D}^{sel}$); our feature-based models are linear models; hyperparameter optimization is done using counterfactual evaluation with clicks on the validation partition [15].

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: clicks are not gathered using the logging policy but by applying the algorithms in an online interactive manner.

5.2 Evaluation

The evaluation metric we use is normalized Discounted Cumulative Gain (DCG) (Eq. 3) [13] using the ground-truth labels from the datasets. To evaluate the high-performance at convergence of tabular models, we do not apply a rank-cutoff when computing the metric; thus, an NDCG of 1.0 indicates that *all* documents are ranked perfectly. Furthermore, we wish to evaluate the performance of GENSPEC on queries with different frequencies. Unfortunately, the datasets do not indicate how frequent each query is. As a solution, we vary the number of total clicks from 100 up to 10^9 in total, uniformly spread over all queries. We separately calculate performance on the test set (Test-NDCG) and the training set (Train-NDCG). Because no clicks are ever generated on the test set, Test-NDCG shows the performance on previously unseen queries. Metrics are always computed over all queries in the partition, thus every reported value of Train-NDCG is based on the entire training set, the same goes for Test-NDCG and the test set. All reported results are averages over 10 runs.

6 RESULTS AND DISCUSSION

6.1 Behavior of GENSPEC

First, we will contrast GENSPEC with a single feature-based model and a single tabular model. Figure 2 shows the performance of (i) GENSPEC with different levels of confidence for its bounds (ϵ), along with that of (ii) the logging policy, (iii) the feature-based model, and (iv) the tabular model between which the GENSPEC chooses. We see that the feature-based model requires few clicks to improve over the logging policy but is not able to reach optimal levels of performance. The performance of the tabular model, 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 tabular model is unable to learn anything and provides random performance (not displayed in Figure 2). 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 [32, 34]; White et al. [42] 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 the tabular model, and never see the improvements it brings at convergence. This possibility can be a large deterrent from applying tabular models in practice [43]. Furthermore, our results indicate there is no simple way to determine when the tabular model is the best choice, i.e., depending on the dataset and the level of noise, this could be after 10^2, 10^3 or more than 10^4 clicks.

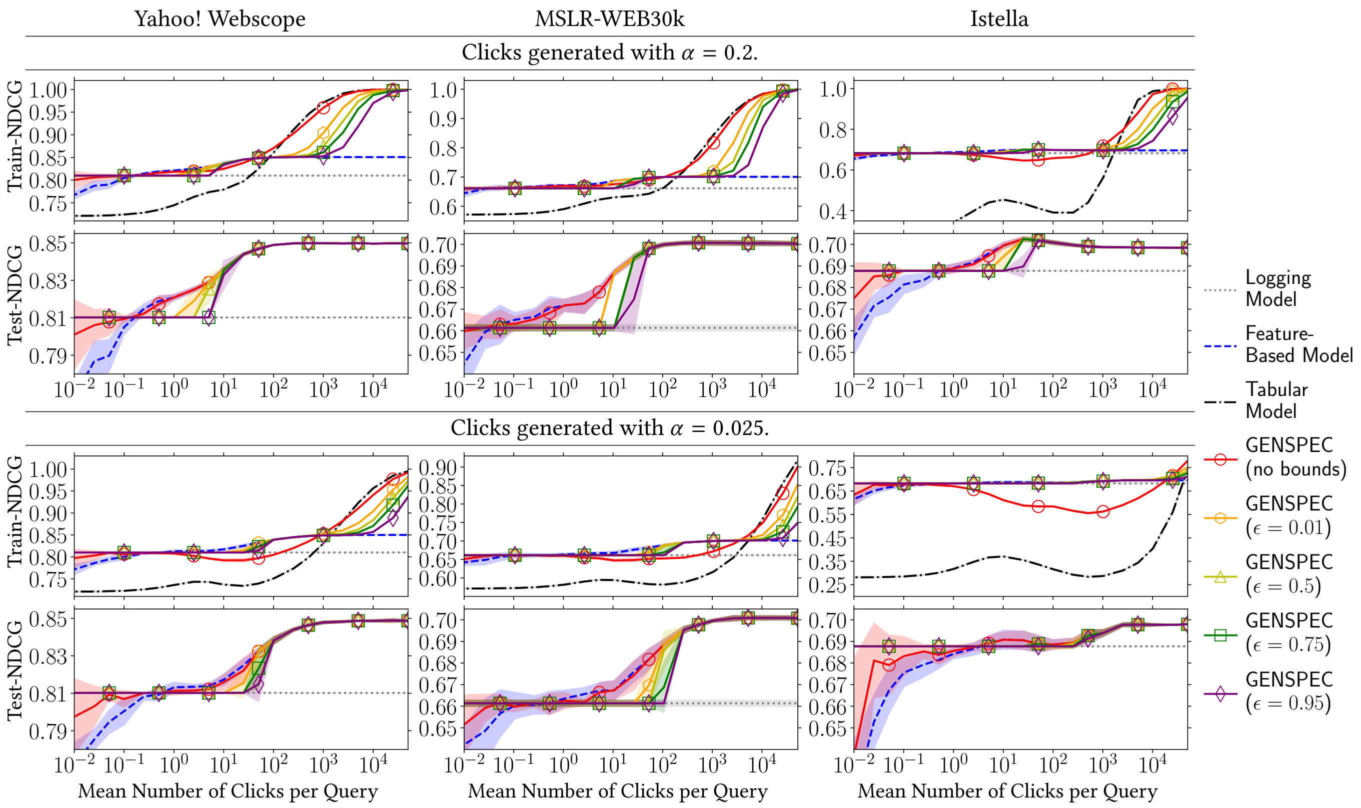

Figure 2: Performance of GENSPEC with varying levels of confidence, compared to pure generalization and pure specialization. We separate queries on the training set (Train-NDCG) that have received clicks, and queries on the test set (Test-NDCG) that do not receive any clicks. Clicks are spread uniformly 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.

Therefore, it is unlikely that a simple heuristic can accurately detect these moments in practice.

Finally, we see that by choosing between the three models GENSPEC combines properties of all: after a few clicks it deploys the feature-based model and thus outperforms the logging policy; as more clicks are gathered, the tabular model is activated on queries, further improving performance. With $\alpha = 0.2$ GENSPEC with $\epsilon \leq 0.75$ reaches perfect Train-NDCG performance on all three datasets by widely deploying the tabular model. However, unlike the tabular model, the performance of GENSPEC (with $\epsilon > 0$) never drops below the logging policy. Moreover, we never observe a situation where an increase in the number of clicks results in a decrease in the mean performance of GENSPEC. As expected, there is a delay between when the tabular model is the optimal choice and when GENSPEC deploys the tabular model on queries. Thus, while the usage of confidence bounds prevents the performance from 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 the tabular model earlier; in some cases these deployments result in worse performance than the logging policy, albeit less than the tabular model on its own. 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 GENSPEC combines the high-performance at convergence of specialization and the safe robustness of generalization. In contrast to tabular models, 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 feature-based models. We only observe a very small delay between when the feature-based model is the optimal choice and when GENSPEC deploys it. We conclude that GENSPEC is generally preferable to pure feature-based counterfactual LTR. Compared to pure tabular counterfactual LTR, GENSPEC is the best choice in situations where periods of poor performance should be avoided [43] or when not all queries receive large numbers of clicks [42].

6.2 Effectiveness of Relative Bounding

To evaluate the efficiency of our relative performance bounds, we apply the SEA bounds due to Jagerman et al. [11] to the GENSPEC strategy. This means that two bounds are used for every choice between two models, each bounding the performance of an individual model. For a fair comparison, we adapt SEA to choose between the same models as GENSPEC and provide it with the same click data.

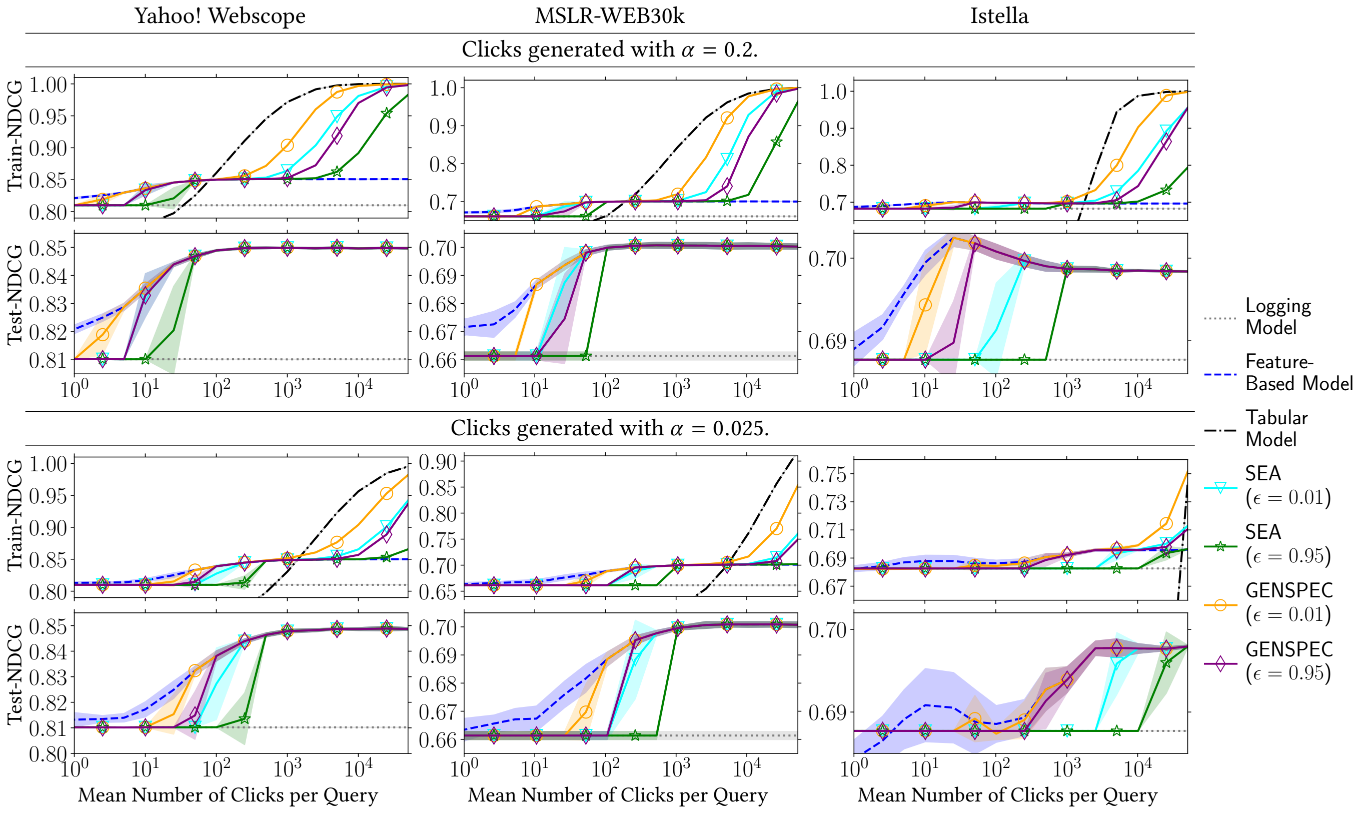

Figure 3: GENSPEC compared to a meta-policy using the SEA bounds (see Section 6.2). Notation is the same as in Figure 2.

Figure 3 displays the results of this comparison. Across all settings, GENSPEC deploys models 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 models 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 (confirming the theory in Appendix B).

6.3 Comparison to Online LTR Bandits

Last, we compared GENSPEC to online bandit LTR algorithms [16, 17]. Unlike counterfactual LTR, these bandit methods learn using online interventions: at each timestep they choose which ranking to display to users. As baselines we use the Hotfix algorithm [46] and the PBM [18]. 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 its randomization is very detrimental to the user experience. In our results, we only report the performance of the ranking produced by the Hotfix baseline, not of the randomized rankings used to gather clicks. We apply two versions of the Hotfix algorithm, one for top-10 reranking and another for 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 [18], which results in PBM always displaying the ranking it expects to perform best, thus attempting to maximize the user experience during learning. These methods all optimize a tabular ranking model: 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, hence, we do not report their Test-NDCG.

Figure 4 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 two reasons: (i) the online interventions of the bandit baselines, and (ii) the delay in deployment added by GENSPEC's usage of confidence bounds. Similar to the tabular model, the earlier 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 [43], or if some queries may not receive large numbers of clicks [42], 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. Overall GENSPEC is thus the safest choice w.r.t. the user experience since it avoids both online interventions and initial periods of poor performance.

7 CONCLUSION

In this paper we have introduced the Generalization and Specialization (GENSPEC) framework for safe query specialization in counterfactual LTR. It simultaneously learns a feature-based model to

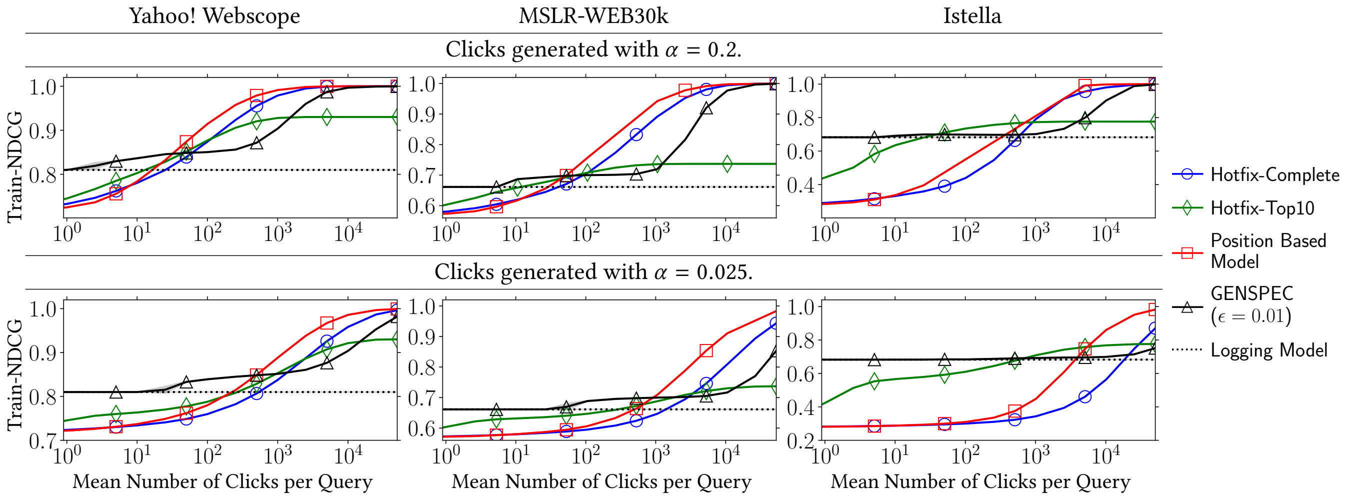

Figure 4: GENSPEC compared to various online LTR bandits (see Section 6.3). Notation is the same as for Figure 2.

perform well across all queries, and a tabular model consisting of many memorized rankings optimized for individual queries. Per query, GENSPEC uses high-confidence bounds to choose between deploying the logging policy, the feature-based model, or the tabular model. Our results show that GENSPEC combines the high performance of a successfully specialized tabular model on queries with sufficiently many interactions, with the safe robust performance of a feature-based model on queries that were previously unseen or where little data is available. As a result, it avoids the low performance at convergence of feature-based models, and the initial poor performance of the tabular models.

We expect GENSPEC to be a framework that will be used for other types of specialization in the future. For instance, we think personalization for LTR is a promising application. Moreover, in Appendix C we describe how GENSPEC could be applied to contextual bandit problems outside of LTR. There are many fruitful directions future work could explore with the GENSPEC framework.

REPRODUCIBILITY

For full reproducibility this work only made use of publicly available data and our complete experimental implementation is publicly available at https://github.com/HarrieO/2021WWW-GENSPEC.

ACKNOWLEDGMENTS

This research was supported by the Netherlands Organisation for Scientific Research (NWO) under project nr 612.001.551. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A General Framework for Counterfactual Learning-to-Rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 5–14.
[2] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019. Addressing Trust Bias for Unbiased Learning-to-Rank. In *The World Wide Web Conference*. ACM, 4–14.
[3] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating Position Bias without Intrusive Interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 474–482.
[4] Qingyao Ai, Tao Yang, Huazheng Wang, and Jiaxin Mao. 2020. Unbiased Learning to Rank: Online or Offline? *arXiv preprint arXiv:2004.13574* (2020).
[5] Christopher J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview.* Technical Report. Microsoft Research.
[6] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. *Journal of Machine Learning Research* 14 (2011), 1–24.
[7] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-bias Models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 87–94.
[8] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. 2016. Fast Ranking with Additive Ensembles of Oblivious and Non-Oblivious Regression Trees. *ACM Transactions on Information Systems (TOIS)* 35, 2 (2016), 1–31.
[9] Trey Grainger. 2021. *AI Powered Search*. Manning.
[10] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. 2013. 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*. ACM, 183–192.
[11] Rolf Jagerman, Ilya Markov, and Maarten de Rijke. 2020. Safe Exploration for Optimizing Contextual Bandits. *ACM Transactions on Information Systems* 38, 3 (2020), Article 24.
[12] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. 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*. ACM, 15–24.
[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.
[14] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 133–142.
[15] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 781–789.
[16] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2016. DCM Bandits: Learning to Rank with Multiple Clicks. In *International Conference on Machine Learning*. 1215–1224.
[17] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. 2015. Cascading bandits: Learning to rank in the cascade model. In *International Conference on Machine Learning*. 767–776.
[18] Paul Lagrée, Claire Vernade, and Olivier Cappé. 2016. Multiple-play Bandits in the Position-based Model. In *Advances in Neural Information Processing Systems*. 1597–1605.
[19] Tor Lattimore and Csaba Szepesvári. 2020. *Bandit Algorithms*. Cambridge University Press.
[20] Chang Li, Branislav Kveton, Tor Lattimore, Ilya Markov, Maarten de Rijke, Csaba Szepesvári, and Masrour Zoghi. 2020. BubbleRank: Safe online learning to re-rank via implicit click feedback. In *Uncertainty in Artificial Intelligence*. PMLR, 196–206.
[21] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[22] Themis Mavridis, Soraya Hausl, Andrew Mende, and Roberto Pagano. 2020. Beyond Algorithms: Ranking at Scale at Booking.com. In *ComplexRec 2020: Workshop on Recommendation in Complex Environments*. ACM.
[23] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable Unbiased Online Learning to Rank. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1293–1302.
[24] Harrie Oosterhuis and Maarten de Rijke. 2019. Optimizing Ranking Models in an Online Setting. In *Advances in Information Retrieval*. Springer International Publishing, Cham, 382–396.
[25] Harrie Oosterhuis and Maarten de Rijke. 2020. 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.
[26] Harrie Oosterhuis and Maarten de Rijke. 2021. Unifying Online and Counterfactual Learning to Rank. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM'21)*. ACM.
[27] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).
[28] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning Diverse Rankings with Multi-armed Bandits. In *Proceedings of the 25th International Conference on Machine Learning*. 784–791.
[29] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. 2008. How Does Clickthrough Data Reflect Retrieval Quality?. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM, 43–52.
[30] Mark Sanderson, Monica Lestari Paramita, Paul Clough, and Evangelos Kanoulas. 2010. Do User Preferences and Evaluation Measures Line Up?. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 555–562.
[31] J Ben Schafer, Joseph Konstan, and John Riedl. 1999. Recommender Systems in e-Commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*. 158–166.
[32] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. 1999. Analysis of a Very Large Web Search Engine Query Log. In *ACM SIGIR Forum*, Vol. 33. ACM New York, NY, USA, 6–12.
[33] Daria Sorokina and Erick Cantu-Paz. 2016. Amazon Search: The Joy of Ranking Products. In *SIGIR*. ACM, 459–460.
[34] Amanda Spink, Seda Ozmutlu, Huseyin C Ozmutlu, and Bernard J Jansen. 2002. US versus European Web Searching Trends. In *ACM Sigir Forum*, Vol. 36. ACM New York, NY, USA, 32–38.
[35] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. 2015. High-confidence Off-policy Evaluation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.
[36] Andrew Trotman, Jon Degenhardt, and Surya Kallumadi. 2017. The Architecture of eBay Search. In *SIGIR Workshop on eCommerce*. ACM.
[37] Ali Vardasbi, Harrie Oosterhuis, and Maarten de Rijke. 2020. When Inverse Propensity Scoring does not Work: Affine Corrections for Unbiased Learning to Rank. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1475–1484.
[38] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A Cascade Ranking Model for Efficient Ranked Retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 105–114.
[39] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 115–124.
[40] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. 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*. ACM, 610–618.
[41] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1313–1322.
[42] Ryen W White, Mikhail Bilenko, and Silviu Cucerzan. 2007. 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*. 159–166.
[43] Yifan Wu, Roshan Shariff, Tor Lattimore, and Csaba Szepesvári. 2016. Conservative Bandits. In *International Conference on Machine Learning*. 1254–1262.
[44] Yisong Yue and Thorsten Joachims. 2009. Interactively Optimizing Information Retrieval Systems as a Dueling Bandits Problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 1201–1208.
[45] Shengyao Zhuang and Guido Zuccon. 2020. Counterfactual Online Learning to Rank. In *European Conference on Information Retrieval*. Springer, 415–430.
[46] Masrour Zoghi, Tomáš Tunys, Lihong Li, Damien Jose, Junyan Chen, Chun Ming Chin, and Maarten de Rijke. 2016. Click-based Hot Fixes for Underperforming Torso Queries. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 195–204.

A PROOF OF UNBIASEDNESS FOR COUNTERFACTUAL LEARNING TO RANK

In this section we will prove that the IPS estimate $\hat{\mathcal{R}}$ (Eq. 9) can be used to unbiasedly optimize the true reward $\mathcal{R}$ (Eq. 4), as claimed in Section 3.2. First, we consider the expected value for an observed click $c_i(d)$ using Eq. 6:

$$\begin{aligned}\mathbb{E}_{y_i,o_i}\big[c_i(d)\big] &= \mathbb{E}_{y_i}\Big[P\big(C=1|o_i(d)=1, r(q_i,d)\big)\cdot P\big(O=1|d,y_i\big)\Big]\\ &= P\big(C=1\mid o_i(d)=1, r(q_i,d)\big)\cdot\Bigg(\sum_{y\in\pi_0}P\big(O=1\mid d,y\big)\cdot\pi_0(y\mid q_i)\Bigg)\\ &= \rho_i(d)\cdot P\big(C=1\mid o_i(d)=1, r(q_i,d)\big).\end{aligned}\tag{32}$$

Note that y_i is the ranking displayed at interaction i while y is the ranking being evaluated, the expected value for the IPS estimator:

$$\begin{aligned}\mathbb{E}_{o_i,y_i}\big[\hat{\Delta}(y\mid c_i,\rho_i)\big] &= \mathbb{E}_{o_i,y_i}\Bigg[\sum_{d\in y}\lambda\big(rank(d\mid y)\big)\cdot\frac{c_i(d)}{\rho_i(d)}\Bigg]\\ &= \sum_{d\in y}\lambda\big(rank(d\mid y)\big)\cdot P\big(C=1\mid o_i(d)=1, r(q_i,d)\big).\end{aligned}\tag{33}$$

This step assumes that $\rho_i(d) > 0$, i.e., that every item has a non-zero probability of being examined [15, 25]. While $\mathbb{E}_{o_i,y_i}[\hat{\Delta}(y\mid c_i,\rho_i)]$ and $\Delta(y\mid q_i,r)$ are not necessarily equal, using Eq. 6 we see that they are proportional with some offset C:

$$\begin{aligned}\mathbb{E}_{o_i,y_i}\big[\hat{\Delta}(y\mid c_i,\rho_i)\big] &\propto \Bigg(\sum_{d\in y}\lambda\big(rank(d\mid y)\big)\cdot r(q_i,d)\Bigg)+C\\ &= \Delta(y\mid q_i,r)+C,\end{aligned}\tag{34}$$

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:

$$\mathbb{E}_{o_i,y_i}\big[\hat{\mathcal{R}}(\pi\mid\mathcal{D})\big]\propto\mathcal{R}(\pi)+C.\tag{35}$$

Consequently, the estimator can be used to unbiasedly estimate the preference between two models:

$$\mathbb{E}_{o_i,y_i}\big[\hat{\mathcal{R}}(\pi_1\mid\mathcal{D})\big] < \mathbb{E}_{o_i,y_i}\big[\hat{\mathcal{R}}(\pi_2\mid\mathcal{D})\big]\Leftrightarrow\mathcal{R}(\pi_1)<\mathcal{R}(\pi_2).\tag{36}$$

Moreover, this implies that maximizing the estimated performance unbiasedly optimizes the actual reward:

$$\arg\max_{\pi}\mathbb{E}_{o_i,y_i}\big[\hat{\mathcal{R}}(\pi\mid\mathcal{D})\big] = \arg\max_{\pi}\mathcal{R}(\pi).\tag{37}$$

This concludes the proof; we have shown that $\hat{\mathcal{R}}$ is suitable for unbiased LTR, since it can be used to find the optimal model.

B EFFICIENCY OF RELATIVE BOUNDING

In this section, we prove that the relative bounds of GENSPEC are more efficient than SEA bounds [11], when the covariance between the reward estimates of two models is positive:

$$\text{cov}\big(\hat{\mathcal{R}}(\pi_1\mid\mathcal{D}),\hat{\mathcal{R}}(\pi_2\mid\mathcal{D})\big)>0.\tag{38}$$

This means that GENSPEC will deploy a model earlier than SEA if there is positive 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 model. Deployment by GENSPEC depends on whether a relative confidence bound is greater than the estimated difference in performance (cf. Eq. 27 and 30). For two models π_1 and π_2 deployment happens when:

$$\hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}) - CB(\pi_1, \pi_2 \mid \mathcal{D}) > 0. \tag{39}$$

Thus the bound has to be smaller than the estimated performance difference:

$$CB(\pi_1, \pi_2 \mid \mathcal{D}) < \hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}). \tag{40}$$

In contrast, SEA does not use a single bound, but two bounds on the individual performances of the models. For clarity, we describe the SEA bound in our notation. First, we have $R^{\pi_j}_{i,d}$, the observed reward for a d at interaction i for model π_j:

$$R^{\pi_j}_{i,d} = \frac{c_i(d)}{\rho_i(d)} \sum_{y \in \pi_j} \pi_j(y \mid q_i) \cdot \lambda\big(rank(d \mid y)\big). \tag{41}$$

Then we have a ν^{π_j} for each model:

$$\nu^{\pi_j} = \frac{2|\mathcal{D}|K \ln\left(\frac{2}{1-\epsilon}\right)}{|\mathcal{D}|K - 1} \sum_{i \in \mathcal{D}} \sum_{d \in y_i} \left(K \cdot R^{\pi_j}_{i,d} - \hat{\mathcal{R}}(\pi_j \mid \mathcal{D})\right)^2,$$

which we use in the confidence bound for a single model π_j:

$$CB(\pi_j \mid \mathcal{D}) = \frac{7Kb \ln\left(\frac{2}{1-\epsilon}\right)}{3(|\mathcal{D}|K - 1)} + \frac{1}{|\mathcal{D}|K} \cdot \sqrt{\nu^{\pi_j}}. \tag{42}$$

We note that the b parameter has the same value for both the relative and single confidence bounds. SEA chooses between model by comparing their upper and lower confidence bounds:

$$\hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - CB(\pi_1 \mid \mathcal{D}) > \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}) + CB(\pi_2 \mid \mathcal{D}). \tag{43}$$

In this case, the summation of the bounds has to be smaller than the estimated performance difference:

$$CB(\pi_1 \mid \mathcal{D}) + CB(\pi_2 \mid \mathcal{D}) < \hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D}). \tag{44}$$

We can now formally describe under which condition GENSPEC is more efficient than SEA: by combining Eq. 40 and Eq. 44, we see that relative bounding is more efficient when:

$$CB(\pi_1, \pi_2 \mid \mathcal{D}) < CB(\pi_1 \mid \mathcal{D}) + CB(\pi_2 \mid \mathcal{D}). \tag{45}$$

We notice that $\mathcal{D}$, K, b and ϵ have the same value for both confidence bounds, thus we only require:

$$\sqrt{\nu} < \sqrt{\nu^{\pi_1}} + \sqrt{\nu^{\pi_2}}. \tag{46}$$

If we assume that $\mathcal{D}$ is sufficiently large, we see that $\sqrt{\nu}$ approximates the standard deviation scaled by some constant:

$$\sqrt{\nu} \approx C \cdot \sqrt{\mathrm{var}\big(\hat{\delta}(\pi_1, \pi_2 | \mathcal{D})\big)}, \text{with } C = \sqrt{\frac{2|\mathcal{D}|^2 K^2 \ln\left(\frac{2}{1-\epsilon}\right)}{|\mathcal{D}|K - 1}}. \tag{47}$$

Since the bounds 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: $\hat{\delta} = \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D})$, $\hat{\mathcal{R}}_1 = \hat{\mathcal{R}}(\pi_1 \mid \mathcal{D})$, and $\hat{\mathcal{R}}_2 = \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D})$. Using the same approximations for $\sqrt{\nu^{\pi_1}}$ and $\sqrt{\nu^{\pi_2}}$ we get:

$$\sqrt{\mathrm{var}(\hat{\delta})} < \sqrt{\mathrm{var}(\hat{\mathcal{R}}_1)} + \sqrt{\mathrm{var}(\hat{\mathcal{R}}_2)}. \tag{48}$$

By making use of the Cauchy-Schwarz inequality, we can derive the following lower bound:

$$\sqrt{\mathrm{var}(\hat{\mathcal{R}}_1) + \mathrm{var}(\hat{\mathcal{R}}_2)} \leq \sqrt{\mathrm{var}(\hat{\mathcal{R}}_1)} + \sqrt{\mathrm{var}(\hat{\mathcal{R}}_2)}. \tag{49}$$

Therefore, the relative bounding of GENSPEC must be more efficient when the following is true:

$$\mathrm{var}(\hat{\delta}) < \mathrm{var}(\hat{\mathcal{R}}_1) + \mathrm{var}(\hat{\mathcal{R}}_2), \tag{50}$$

i.e., the variance of the relative estimator must be less than the sum of the variances of the estimators for the individual model. Finally, by rewriting $\mathrm{var}(\hat{\delta})$ to:

$$\mathrm{var}(\hat{\delta}) = \mathrm{var}(\hat{\mathcal{R}}_1 - \hat{\mathcal{R}}_2) = \mathrm{var}(\hat{\mathcal{R}}_1) + \mathrm{var}(\hat{\mathcal{R}}_2) - 2\mathrm{cov}(\hat{\mathcal{R}}_1, \hat{\mathcal{R}}_2), \tag{51}$$

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:

$$\mathrm{cov}(\hat{\mathcal{R}}_1, \hat{\mathcal{R}}_2) > 0. \tag{52}$$

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.

C 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 $z_i \sim P(Z)$. Data is gathered using the logging policy π_0: $a_i \sim \pi_0(a \mid z_i)$, where a indicates an action. However, unlike the LTR case, the rewards r_i are observed directly: $r_i \sim P(r \mid a_i, z_i)$. With the propensities $\rho_i = \pi_0(a_i \mid z_i)$ the data is: $\mathcal{D} = \{(r_i, a_i, \rho_i, z_i)\}_{i=1}^N$; for specialization the data is filtered per context z: $\mathcal{D}_z = \{(r_i, a_i, \rho_i, z_i) \in \mathcal{D} \mid z_i = z\}$. Again, data for training $\mathcal{D}^{train}$ and for model selection $\mathcal{D}^{sel}$ are separated. The reward is estimated with an IPS estimator:

$$\hat{\mathcal{R}}(\pi \mid \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \frac{r_i}{\rho_i} \pi(a_i \mid z_i). \tag{53}$$

Again, we have a policy trained for generalization π_g and another for specialization π_z, and the copies trained on $\mathcal{D}^{train}$: π_g' and π_z'. The difference between two policies is estimated by: $\hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) = \hat{\mathcal{R}}(\pi_1 \mid \mathcal{D}) - \hat{\mathcal{R}}(\pi_2 \mid \mathcal{D})$. We differ from the LTR approach by estimating the bounds using:

$$R_i = \frac{r_i}{\rho_i} \big(\pi_1(a_i \mid x_i, z_i) - \pi_2(a_i \mid x_i, z_i)\big). \tag{54}$$

Following Thomas et al. [35], the confidence bounds are:

$$\begin{aligned} CB(\pi_1, \pi_2 \mid \mathcal{D}) = {} & \frac{7b \ln\left(\frac{2}{1-\epsilon}\right)}{3(|\mathcal{D}| - 1)} \\ & + \frac{1}{|\mathcal{D}|} \sqrt{\frac{2|\mathcal{D}| \ln\left(\frac{2}{1-\epsilon}\right)}{|\mathcal{D}| - 1} \sum_{i \in \mathcal{D}} \left(R_i - \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D})\right)^2}, \end{aligned} \tag{55}$$

where b is the maximum possible value for R_i. This results in the lower bound $LCB(\pi_1, \pi_2 \mid \mathcal{D}) = \hat{\delta}(\pi_1, \pi_2 \mid \mathcal{D}) - CB(\pi_1, \pi_2 \mid \mathcal{D})$. GENSPEC first chooses between the logging policy and π_g:

$$\pi_G(a \mid z) = \begin{cases} \pi_g(a \mid z), & \text{if } LCB(\pi_g', \pi_0 \mid \mathcal{D}^{sel}) > 0 \\ \pi_0(a \mid z), & \text{otherwise,} \end{cases} \tag{56}$$

and then per context z whether the policy π_z will be activated:

$$\pi_{GS}(a \mid z) = \begin{cases} \pi_z(a \mid z), & \text{if } LCB(\pi_z', \pi_G' \mid \mathcal{D}_z^{sel}) > 0 \\ \pi_G(a \mid z), & \text{otherwise.} \end{cases} \tag{57}$$

As such, GENSPEC can be applied to the contextual bandit problem for any arbitrary choice of context z.