Can clicks be both labels and features? Unbiased Behavior Feature Collection and Uncertainty-aware Learning to Rank

Tao Yang
University of Utah
Salt Lake City, Utah, USA
taoyang@cs.utah.edu

Chen Luo
Amazon Search
Palo Alto, CA, USA
cheluo@amazon.com

Hanqing Lu
Amazon Search
Palo Alto, CA, USA
luhanqin@amazon.com

Parth Gupta
Amazon Search
Palo Alto, CA, USA
guptpart@amazon.com

Bing Yin
Amazon Search
Palo Alto, CA, USA
alexbyin@amazon.com

Qingyao Ai
University of Utah
Salt Lake City, Utah, USA
aiqy@cs.utah.edu

ABSTRACT

Using implicit feedback collected from user clicks as training labels for learning-to-rank algorithms is a well-developed paradigm that has been extensively studied and used in modern IR systems. Using user clicks as ranking features, on the other hand, has not been fully explored in existing literature. Despite its potential in improving short-term system performance, whether the incorporation of user clicks as ranking features is beneficial for learning-to-rank systems in the long term is still questionable. Two of the most important problems are (1) the explicit bias introduced by noisy user behavior, and (2) the implicit bias, which we refer to as the exploitation bias, introduced by the dynamic training and serving of learning-to-rank systems with behavior features. In this paper, we explore the possibility of incorporating user clicks as both training labels and ranking features for learning to rank. We formally investigate the problems in feature collection and model training, and propose a counterfactual feature projection function and a novel uncertainty-aware learning to rank framework. Experiments on public datasets show that ranking models learned with the proposed framework can significantly outperform models built with raw click features and algorithms that rank items without considering model uncertainty.

CCS CONCEPTS

• **Information systems → Learning to rank**.

KEYWORDS

Learning to rank, Behavior feature, Exploitation bias

1 INTRODUCTION

Ranking is a core component of many Information Retrieval (IR) applications. Among different types of ranking techniques, learning to rank (LTR) [36], which ranks items by building ranking functions with machine learning (ML) models, is one of the most popular ranking frameworks in modern IR systems. With recent advances on ML models such as gradient boosting machines [27] and deep learning techniques [30], LTR algorithms have dramatically improved the retrieval performance of search engines and recommendation systems. However, this also increases the need for labeled data. As collecting large-scale explicit relevance annotations is expensive and prohibitive in many applications, implicit feedback extracted from user behavior data has been widely used for training LTR models [33]. For example, previous studies [5, 34, 59] have shown that effective LTR models can be learned directly from training labels constructed with user clicks and unbiased learning techniques.

As user clicks have been well recognized as good alternatives for explicit relevance labels in practice [61], a natural question is: *can we use user clicks as ranking features for LTR models as well*? In fact, many industrial IR systems have already considered user clicks as important input signals for their LTR models [14]. For example, Agichtein et al. [3] have shown that, by incorporating user clicks as behavior features to ranking systems, they can improve the performance of competitive web search ranking algorithms by as much as 31% relative to the original performance. Because user behaviors are direct indicators of result utility from the user's perspective [59], theoretically speaking, no other features can reflect result relevance better than the behavior signals collected from the user themselves.

Nonetheless, whether the incorporation of user behavior features would benefit or hurt the overall quality of LTR systems is still controversial. While the superior performance of LTR models with behavior features has been broadly observed in ranking experiments with short-term evaluation metrics (e.g., nDCG and click-through rates), it is also recognized that directly using user clicks as ranking features could bring destructive damage to retrieval quality in the long term. To the best of our knowledge, there are two reasons why incorporating user behavior features could be problematic in practice.

First, user clicks are noisy. User studies have found that web search users tend to examine and click results on top of search engine result pages (SERPs) despite the actual utility of those results (i.e., the position bias [33] and trust bias [60]). Without proper treatment, incorporating clicks as features will enhance the effect of click bias on LTR models and eventually drive them to rank items according to their previous rankings but not their actual relevance.

SIGIR'22, July 11-15, 2022, Madrid, Spain
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The second reason, which is more important, is that the incorporation of clicks as features would create a new type of bias into the input data directly. For example, because behavior features usually have high correlations with the training labels in LTR (especially when we use user clicks as the training labels), such features could easily overwhelm other features in training and dominate model outputs. In practice, behavior features can be collected only for old items that have been shown to users previously, and we cannot extract the same feature accurately for new items since we haven't explore and show them to users yet. Without proper treatments, the training of LTR models with behavior features could easily overexploit known information and produce ranking models that always prefer old items over new ones. This makes new items more difficult to be explored and creates a vicious spiral that forever prevents them from being shown to users. While certain bias in LTR labels (e.g., selection bias [39, 59]) could be alleviated with post-processing or non-behavior features (as shown in Section 6.2.3), the above bias in ranking features, which we refer to as the *exploitation bias*, is a more fundamental problem that is coupled with the dynamic training and serving of LTR systems.

In this paper, we address the problems above with new feature collection and ranking exploration algorithms for LTR. Specifically, inspired by unbiased learning to rank [2, 55], we first propose to resolve the noise in click features with counterfactual affine functions and discuss multiple paradigms to incorporate unbiased behavior features in LTR systems. After that, we propose to address the exploitation bias in LTR with a novel uncertainty-aware learning to rank algorithm. Collecting click features for previously unseen items requires exploration in ranking, but myopically exploring new items is also undesirable because (1) it could significantly hurt user experience, and (2) there is no trivial method to decide whether an item is well-explored or still under-explored. Based on these observations, we propose to conduct ranking exploration based on the uncertainty of LTR models on each candidate item. The intuition is that we should explore an item and let users interact with it when our model is uncertain about the item's utility, and gradually reduce its chance of exploration when we gain more confidence in our prediction. To demonstrate the effectiveness of the proposed algorithms, we conducted experiments on public LTR datasets with synthetic clicks. We simulated the process of click feature collection and new item discovery to evaluate the long-term ranking performance of each method. Experiments show that our unbiased feature affine functions and uncertainty-aware learning-to-rank algorithm can significantly outperform both baselines that use raw clicks as features and models that conduct exploration without considering uncertainty in ranking.

2 RELATED WORK

In general, there are three lines of research that directly relate to this paper, that are behavior features in LTR, unbiased/online learning to rank, and uncertainty in ranking.

Behavior features in LTR. User behavior signals have been treated as important indicators for result relevance in IR [10]. With the emergence of the internet and large-scale Web search engines in 1990s, using user behaviors and feedback to improve retrieval performance has gradually become a popular paradigm for modern IR systems. For example, Agichtein et al. [3] incorporated user clicks as features and significantly improved the ranking performance of a commercial IR system. Ferro et al. [25, 26] and Macdonald et al. [37] showed that user behavior features can significantly improve upon an effective learned model without behavior features. Usta et al. [54] also showed that ranking models trained with features based on user behavior can outperform various baselines based on adhoc retrieval functions on education search engines. Nonetheless, despite the popularity of behavior features, few studies have formally discussed the effect of user clicks as behavior features in LTR. As shown in this paper, incorporating behavior features without proper treatments could hurt the effectiveness of LTR systems by amplifying the problem of overexploitation and overfitting. Some strategies were proposed to alleviate the problem by predicting behavior features with non-behavior features [31]. Unfortunately, these methods cannot address the problem because the predictions usually suffer from significant errors, and the models they create are theoretically the same to the LTR models without behavior features, which were shown to be suboptimal [3, 25, 26].

Unbiased and Online LTR. In contrast to the discussion of how to use clicks as ranking features, the analysis of how to use user clicks as training labels for LTR, on the other hand, has been extensively studied in the last decade [6, 7, 53]. The major focus of these studies is how to effectively learn unbiased LTR models by training them with biased click signals [32, 33]. Specifically, the studies of how to actively remove biases in labels through online interpolations (i.e., online LTR) and how to address click bias from theoretical perspectives (i.e., unbiased LTR) have received considerable attention. For example, previous studies have developed different strategies to explore result relevance with bandit learning [48, 49, 56–58, 63] or stochastic ranking sampling [38] in online LTR systems. To train LTR models with offline click logs, causal analysis techniques such as counterfactual learning [1, 4, 34, 40, 62] have also been widely adopted in extracting unbiased training objectives for LTR. In contrast to existing studies that only treat user clicks as training labels in unbiased/online LTR, in this paper, we explore and analyze the effect of using click data as both features and labels for LTR.

Uncertainty in Ranking. Model uncertainty has been widely studied in the community of ML and statistics [16, 23, 28]. In IR, one of the first studies that use model uncertainty for ranking is proposed by Zhu et al. [64], in which they use the variance of a probabilistic language model as a risk-based factor to improve the performance of retrieval models. Instead of optimizing ranking performance directly, there are also studies that use model uncertainty to improve query performance prediction [46, 50] and query cutoff prediction [19, 35]. Recently, as neural retrieval models have become popular in modern IR systems, uncertainty estimation techniques for deep learning models have been introduced into the studies of neural IR [17, 42]. It has been shown that model uncertainty in neural networksc can help us better understand and analyze the behaviors of neural rankers, such as BERT-based models [21]. In contrast to previous studies, in this paper, we propose to use model uncertainty for ranking exploration in LTR. Instead of optimizing short-term ranking metrics directly, our proposed uncertainty-aware LTR algorithm can collect high quality behavior features and build effective ranking models at the same time.

3 PROBLEM FORMULATION

In this section, we formally describe the problem of learning to rank with implicit user feedback as well as the hypothesis we used to build and analyze the proposed methods in theory. Specifically, let q be a query, d be a candidate document/item to be ranked, and x be the feature vector of (q, d). Given a specific dataset Q, the goal of learning to rank is to construct a ranking function f (parameterized by θ) that minimizes a ranking loss $\mathcal{L}(Q)$ defined over the predicted relevance of each document, i.e., $f(x|d, q, \theta)$, and the actual relevance of each document, i.e., $P(R = 1|q, d)$, as

$$\mathcal{L}(Q) = \frac{1}{|Q|} \sum_{q \in Q} \sum_{d \in \pi_q} l(f(x|d, q, \theta), P(R = 1|q, d)) \tag{1}$$

where π_q is the ranked list of documents created by $f(x|d, q, \theta)$. Depending on how we evaluate the quality of ranked lists, the local loss function l can be defined with different weighting schemes according to the relevance and position of the document in π_q.

Ideally, $P(R = 1|q, d)$ should be inferred from explicit user feedback such as relevance annotations, but this is often infeasible due to the prohibitive costs of collecting large-scale explicit feedback. Therefore, a popular way to address the problem is to replace explicit feedback with implicit feedback such as user clicks to build and train learning-to-rank models. However, user clicks often suffer from several types of noise and biases in practice, the most well-known two of which are the position bias [18] and the trust bias [33]. The position bias refers to the tendency of users towards examining documents at higher positions. And the trust bias describes the tendency of users towards trusting the quality of ranking systems and clicking the results without checking about their actual relevance. Formally, let C and E be the Bernoulli variables representing whether a user has clicked or examined a document, respectively. Following a common examination hypothesis used in previous studies [2, 55], the probability of a document being clicked can be formulated as

$$\begin{aligned} P(C{=}1|d, q, k) = P(E{=}1|k)\Big(&P(C{=}1|R{=}1, E{=}1)P(R{=}1|d, q) \\ &+ P(C{=}1|R{=}0, E{=}1)P(R{=}0|d, q)\Big) \end{aligned} \tag{2}$$

where k is the position of d in q when presented to users. Fortunately, many methods have already been proposed to estimate position bias and trust bias in practice [2, 5, 34, 55, 59]. Thus, for the simplicity of theoretical analysis, we assume that $P(E{=}1|k)$, $P(C{=}1|R{=}1, E{=}1)$, and $P(C{=}1|R{=}0, E{=}1)$ are known in advance.

Similar to previous studies [7, 33], the goal of this paper is to construct LTR models directly from user clicks to improve relevance ranking. However, to the best of our knowledge, most existing studies on unbiased and online learning to rank with implicit feedback only use user clicks as supervision signals (i.e., labels) [34, 38–40, 59]. They simply assume that the feature vectors of documents (i.e., x) are static and extracted independently of the labels. In contrast, we believe that a natural paradigm in practice is to incorporate click signals as a part of the ranking features for LTR directly. We refer to the static and click-independent features as *non-behavior features* (i.e., x_{nb}) and the features constructed from user clicks as *behavior features* (i.e., x_b). Our goal is to construct better LTR models by simultaneously using click data as labels and features.

4 REPRESENTING CLICKS AS FEATURES

We now describe how to collect and use user clicks as behavior features for LTR. In contrast to non-behavior features such as term matching scores [45], behavior features are dynamically collected in online services and subject to significant user noise. Therefore, how to design and use click-based features is an important problem in modern IR systems. In this work, we collect click features dynamically during the training and serving of LTR systems. Inspired by the studies of unbiased learning to rank, we propose a theoretically principled method to debias click features and discuss different paradigms to incorporate them into LTR frameworks.

4.1 Click Collection and System Update

Previous studies [3, 44] often assume that behavior features are collected beforehand and independent of the construction of LTR models. This assumption is problematic because, in real-world applications, LTR systems can directly control the distribution of results presented to users and significantly affect when and where we can collect click data. Therefore, in this paper, we formulate the problem with a more realistic setting by assuming that click features are dynamically collected together with the training and serving of LTR systems.

Suppose that our system receives and processes one query session at each time step. At each time step t, the user issues query q_t, and then the ranking system selects documents from candidate set D_{q_t}, displays ranking π_t, and collects clicks c_t on π_t. Since updating features and ranking models at each time step is often prohibitive in practice [7], a common paradigm to develop LTR systems is to first serve an initial LTR model to users, collect user interactions for a time period, and then use the collected clicks to update the online system. Assuming that the LTR system will be updated at time step T, we need to extract behavior features and update models with a set of click data collected over T time steps as

$$\mathcal{D} = \{(q_t, D_{q_t}, \pi_t, c_t)\}_{t=1}^{T} \tag{3}$$

Note that even for the same query/user, the candidate set D_{q_t} and π_t could change from time to time when new documents are introduced to the database. By formulating the problem in this way, we aim to better analyze and model data dynamics in practical ranking scenarios.

4.2 Unbiased Behavior Feature Extraction

One of the most straightforward methods to extract click-based behavior features is to compute the probability of clicking a document with its cumulated clicks and impressions, e.g., clickthrough rate (CTR). However, the naive CTR of a document doesn't directly reflect document relevance due to the biases in click data. For example, we can derive the CTR of a (q, d) pair from Eq. (2) as

$$P(C = 1|d, q, k) = \alpha_k P(R = 1|d, q, k) + \beta_k \tag{4}$$

where we have

$$\begin{aligned} \alpha_k &= P(E{=}1|k) * (P(C{=}1|R{=}1, E{=}1) - P(C{=}1|R{=}0, E{=}1)) \\ \beta_k &= P(E{=}1|k) * (P(C{=}1|R{=}0, E{=}1)) \end{aligned} \tag{5}$$

Because α_k and β_k could vary significant based on the document position k, $P(C = 1|d, q, k)$ cannot be used as a direct replacement of $P(R = 1|d, q, k)$. This means that the naive CTR of documents is a biased estimation of document relevance.

Inspired by recent advances in unbiased learning to rank [2, 34, 55], we propose to adopt an affine function over clicks to extract unbiased click features. Specifically, given $\mathcal{D}$, the probability of a document being clicked in spite of its position can be computed as

$$\begin{aligned} P(C=1|d,q) &= \frac{1}{T_d} \sum_{\substack{t=1 \\ q_t=q, d\in\pi_t}}^{T} \left(\alpha_{Rnk(d,\pi_t)} P(R=1|d,q) + \beta_{Rnk(d,\pi_t)}\right) \\ &= \frac{1}{T_d}\left(Cum^T[\alpha_{d,q}]P(R=1|d,q) + Cum^T[\beta_{d,q}]\right) \end{aligned} \quad (6)$$

where $Rnk(d, \pi_t)$ is the position of d in the ranked list π_t, and

$$\begin{aligned} T_d = \sum_{\substack{t=1 \\ q_t=q,d\in\pi_t}}^{T} 1, \qquad Cum^T[\alpha_{d,q}] = \sum_{\substack{t=1 \\ q_t=q,d\in\pi_t}}^{T} \alpha_{Rnk(d,\pi_t)} \\ Cum^T[\beta_{d,q}] = \sum_{\substack{t=1 \\ q_t=q,d\in\pi_t}}^{T} \beta_{Rnk(d,\pi_t)} \end{aligned} \quad (7)$$

As discussed in previous studies [2, 59], α and β can be estimated through online experiments in advance. Thus, following the idea of affine estimators proposed by Vardasbi et al. [55], we can extract behavior feature x_b at time T with an affine click probability of d as

$$\Delta^T(d|q) = \frac{Cum^T[C_{d,q}] - Cum^T[\beta_{d,q}]}{Cum^T[\alpha_{d,q}]}, Cum^T[C_{d,q}] = \sum_{\substack{\tau=1 \\ q_t=q,d\in\pi_\tau}}^{T} c_\tau(d) \quad (8)$$

where T is the most recent feature updating time step. In practice, updating ranking systems in real time is often prohibitive. Therefore, we assume that features and models are updated periodically and the behavior feature x_b computed at T is fixed and used in future time steps till the next periodic system update.

The affine click probability $\Delta^T(d|q)$ is an unbiased estimation of $P(R=1|d, q)$ given $\mathcal{D}$ as

$$\begin{aligned} &\mathbb{E}_c[\Delta^T(d|q)] = \frac{\mathbb{E}_c\left[Cum[C_{d,q}]\right] - Cum[\beta_{d,q}]}{Cum[\alpha_{d,q}]} \\ &= \frac{\mathbb{E}_c\left[\sum^T_{\substack{t=1 \\ q_t=q,d\in\pi_t^*}} c_{Rnk(d,\pi_t)}\right] - Cum[\beta_{d,q}]}{Cum[\alpha_{d,q}]} \\ &= \frac{\sum^T_{\substack{t=1 \\ q_t=q,d\in\pi_t^*}} P(C=1|d,q,\pi_t) - Cum[\beta_{d,q}]}{Cum[\alpha_{d,q}]} \\ &= \frac{\sum^T_{\substack{t=1 \\ q_t=q,d\in\pi_t^*}} \left(\alpha_{Rnk(d,\pi_t)}P(R=1|d,q) + \beta_{Rnk(d,\pi_t)}\right) - Cum[\beta_{d,q}]}{Cum[\alpha_{d,q}]} \\ &= \frac{Cum[\alpha_{d,q}]P(R=1|d,q) + Cum[\beta_{d,q}] - Cum[\beta_{d,q}]}{Cum[\alpha_{d,q}]} \\ &= P(R=1|d,q) \end{aligned} \quad (9)$$

Note that the above equation is satisfied when $Cum^T[\alpha_{d,q}] > 0$, which means that d has been shown to users in $\mathcal{D}$. However, it is unrealistic to expect every document to have $Cum^T[\alpha_{d,q}] > 0$ when we can only show a limited number of documents in each session. Thus, we set a default value for $\Delta^T(d|q)$ (e.g., -2)[1] when $Cum^T[\alpha_{d,q}] = 0$.

In contrast to the relevance estimator constructed based on how documents and clicks were collected in the previous ranking system (i.e., the logging policy) [40], our affine estimator is logging-policy-oblivious since it requires no prior knowledge on the logging policy. This is particularly important because, in many cases, we don't have a stationary logging policy in online systems. For example, when new documents are constantly introduced into the candidate sets, the probability of giving the same ranked list could be different at different time steps. As shown in Section 6, our models with affine click features can significantly outperform models with naive CTR despite the updates of ranking models and the introduction of new documents in online inferences.

4.3 Behavior Feature Incorporation

The final step of feature processing is feeding the features to an actual LTR system and building models accordingly. In this paper, we discuss three paradigms for behavior feature incorporation in LTR, namely *ranking without behavior features*, *user feedback as regular features*, and, *user feedback as independent evidence*.

4.3.1 *Ranking without Behavior Features.*

The first paradigm, which is also one of the most popular paradigms used in academic studies, is to build LTR models without any features extracted from user interactions. In other words, the ranking function f would be built purely based on non-behavior features x_{nb} such as query-document similarity scores (e.g., BM25), document quality scores (e.g., PageRank) [43], etc. Formally, we refer to the model built with $\mathcal{D}$ under this paradigm as

$$f_{w/o}(d, q, \theta^T) = f(x_{nb}|d, q, \theta^T) \quad (10)$$

where θ^T is the parameter of f after training with $\mathcal{D}$. The advantages of such a paradigm is that all the features can be extracted beforehand and fixed during the training and serving of LTR models. However, as our goal is to investigate the possibility of using behavior features for LTR, this paradigm is mainly served as a baseline in this paper.

4.3.2 *User Feedback as Regular Features.*

The second paradigm to incorporate click features in LTR is to treat them in the same way with other regular features. Formally, we have

$$f_R(d, q, \theta^T) = f([x_{nb}, x_b]|d, q, \theta^T) \quad (11)$$

where $[x_{nb}, x_b]$ represents the concatenation of non-behavior features x_{nb} and behavior feature x_b. As most existing LTR models such as regression trees [15] and neural networks [4, 11, 12] have the ability to automatically adapt and use input features without knowing their structures and meanings, this paradigm is one of the most natural methods to extend LTR models with new features. It has been widely used in IR industry. Specifically, we could either use CTR or $\Delta^T(d|q)$ as x_b in Eq. (11).

[1]Note that $\Delta^T(d|q)$ could be negative and -2 is smaller than the minimum value of $\Delta^T(d|q)$ that we have observed in our experiments.

4.3.3 User Feedback as Independent Evidence. While the incorporation of behavior features as regular features is appealing, it could also introduce severe over-fitting problems to model training when click data are used to extract both labels and features for LTR. For example, previous studies on unbiased learning to rank derive loss functions directly from click data. When we use clicks as behavior features, x_b could easily dominate the training of $f_R(d, q, \theta^T)$ (due to its high correlation to the training labels) and make x_{nb} look useless in many cases. Such phenomenon is particularly harmful when we rank new documents or documents that are underexplored in historical data because these documents may not have x_b in advance. To address this problem, inspired by the idea of directly re-ranking documents with user interactions [3], we propose to treat x_b as independent ranking evidence and combine it with LTR models built on non-behavior features as

$$f_{IE}(d, q, \theta^T) = \begin{cases} \Delta^T(d, q), & \text{if } Cum^T[\alpha_{d,q_t}] > 0 \\ f(x_{nb}|d, q, \theta^T), & \text{if } Cum^T[\alpha_{d,q_t}] = 0 \end{cases} \tag{12}$$

The motivation is to rank documents that haven't been shown to users with non-behavior features so that they won't be punished due to the cold start problem. Once we collect more user interactions on the documents, we directly rank them with $\Delta^T(d, q)$ since it is an unbiased estimation of $P(R=1|d, q)$. For simplicity, we directly train f to predict the estimated document relevance (as discussed in Section 6.1.4) so that $f(x_{nb}|d, q, \theta^T)$ and $\Delta^T(d, q)$ are comparable. We leave more advanced methods to train f for future studies.

5 UNCERTAINTY-AWARE LEARNING TO RANK

While the utilization of unbiased learning techniques can alleviate the problem of position bias and trust bias in click features, there is another type of bias that remains unsolved in behavior feature collection, which we refer to as the *exploitation bias*. Exploitation bias is a type of data bias introduced by the dynamic training and serving LTR systems. A similar concept frequently used in the studies of unbiased learning to rank is the selection bias [59]. The selection bias focuses on biased label distributions created by user behaviors and system interfaces, i.e., user clicks cannot be collected on documents lower than certain ranks due to the limited display space of user interfaces. As shown in Section 6.2, such bias can be effectively reduced with unbiased LTR techniques and ranking features with click-independent feature distributions. In contrast, exploitation bias focuses on biased data distributions that appear when we use click-dependent features in LTR. When we use clicks collected on previous ranking systems to extract behavior features, documents explored by old online systems will receive significant advantages in terms of feature collection compared to other documents. This bias in input feature distribution will inevitably affect the training of LTR systems despite of whether we have unbiased training labels or not.

In order to address exploitation bias, online LTR systems need to guarantee that all candidate documents, no matter old or new, should be explored in a certain amount of sessions so that they can collect enough user feedback to extract reliable behavior features. Specifically, there are two key problems: (1) what documents we should explore in each session, and (2) how we explore those documents by balancing the quality of feedback collection and the performance of online systems. In this section, we propose an uncertainty-aware learning-to-rank framework to address the exploitation bias. The basic idea is that we should explore a document when we are uncertain about its relevance. After we gradually collect more user feedback on the document and have more confidence in our prediction, the exploration rate should be reduced in order to guarantee the overall quality of online services. To achieve these goals, we propose to conduct ranking exploration with the upper confidence bound selection and discuss how to estimate ranking uncertainty with different ranking paradigms.

5.1 Exploration with Upper Confidence Bound

Upper Confidence Bound (UCB) selection [8, 9] refers to a classic exploration algorithm that have been widely used in bandit learning [13, 29]. Intuitively, UCB follows the principle of optimism in the face of uncertainty which implies that if we are uncertain about an action, we should optimistically assume that it is the correct action [47]. In ranking problems, this indicates that, when we are uncertain about a document, we should explore it by using the upper confidence bound of its predicted score for ranking.

Formally, suppose that we have trained and updated the online system with the clicks and features extracted from dataset $\mathcal{D}$ at time step T. Let $es_t(d, q)$ be the estimated uncertainty of d in q, then we propose to conduct ranking exploration with the framework of UCB as the followings:

$$s_t(d, q_t) = f(d, q_t, \theta^T) + \lambda es_t(d, q_t), \text{ for } t > T \tag{13}$$

where $f(d, q_t, \theta^T)$ is the original ranking score predicted by LTR models, and λ is a hyper-parameter that controls the weights of exploration. The final ranked list shown to users is computed as

$$\pi_t = \arg_{topK}\left(s_t(d, q_t) \in D_{q_t}\right) \tag{14}$$

where $\arg_{topK}$ means sorting and selecting the top K documents to from π_t. In this paper, we assume that each session can only show at most K documents to users. By sorting documents according to both their ranking scores and uncertainty, we can explore documents based on our confidence in their relevance. As shown in Section 6.2, this can help us efficiently reduce the exploitation bias in behavior features and LTR models. The only remaining question is how to estimate the uncertainty of each (q, d) pair.

5.2 Uncertainty Estimation in Ranking

A well-established method to estimate uncertainty in UCB is to use the variance of model outputs. In ranking problems, we can treat the predicted ranking score of each document as model outputs and estimate the uncertainty of each (q, d) pair as

$$es_t(d, q) = std\big(f(d, q_t, \theta^T)\big) \tag{15}$$

where std is the function of standard deviation. Depending on how we derive $f(d, q_t, \theta^T)$ in Section 4.3, we propose different methods to estimate the standard deviation of ranking outputs in this section.

5.2.1 Model-based uncertainty estimation. As most LTR functions are directly implemented with machine learning models, the most straightforward method to estimate uncertainty in ranking is to compute the standard deviation of the machine learning models.

Previous studies have developed a variety of algorithms to compute the standard deviation of model output for different ML models [22, 24]. For example, since many neural models use dropout [52] as efficient regularization for parameter learning, a common method to estimate the variance of neural ranking models [17, 42] is to adapt Monte Carlo Dropouts [28]. Intuitively, during the inference of a (q, d) pair, we can conduct Monte Carlo dropouts on the last layer of neural networks by dropping neuron outputs with factor z_θ (randomly sampled from a uniform Bernoillis distribution) for N times and get N different output scores $\{f(d, q_t, z^i_\theta, \theta^T)\}$ (for i from 1 to N). The standard deviation of $f(d, q_t, \theta^T)$ can then be computed as

$$std\big(f(d, q_t, \theta^T)\big) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\big(f(d, q_t, z^i_\theta, \theta^T) - \mathbb{E}[f]\big)^2} \tag{16}$$

where $\mathbb{E}[f] = \frac{1}{N}\sum_{i=1}^{N} f(d, q_t, z^i_\theta, \theta^T)$. For simplicity, we implement all ranking models in this paper with deep neural networks, so Monte Carlo dropouts can be directly used to estimate the uncertainty of $f_{w/o}(d, q_t, \theta^T)$ in Eq. (10) and $f_R(d, q_t, \theta^T)$ in Eq. (11). We leave the discussion of other types of ranking models with other uncertainty estimation methods for future studies.

5.2.2 Interaction-based uncertainty estimation. When using behavior features as independent evidence (e.g., Section 4.3.3), the uncertainty of model outputs involve two separate parts: the uncertainty of ranking models based on non-behavior features (i.e., $f(x_{nb}|d, q_t, \theta^T)$), and the uncertainty of behavior evidence $\Delta^T(d, q_t)$ built from user interactions. The standard deviation of $f(x_{nb}|d, q_t, \theta^T)$ can be extracted easily based on model-based uncertainty estimation methods described above. Here we only discuss how to derive the uncertainty of $\Delta^T(d, q_t)$.

For simplicity, let $Cum[C] = Cum[C_{d,q}]$, $Cum[\beta] = Cum[\beta_{d,q}]$, $Cum[\alpha] = Cum[\alpha_{d,q}]$, and $C_i = C_{d,\pi}$. According to Eq. (9), we have

$$\begin{aligned}\mathbb{E}_c[\Delta^T(d, q_t)]^2 &= P(R = 1|d, q)^2 \\ &= \frac{\mathbb{E}_c[Cum[C]]^2 - 2\,\mathbb{E}_c[Cum[C]]Cum[\beta] + Cum[\beta]^2}{Cum[\alpha]^2}\end{aligned} \tag{17}$$

Also, we have

$$\begin{aligned}\mathbb{E}_c[Cum[C]^2] &= \mathbb{E}_c[\sum_i^{T_d} C_i + \sum_j^{T_d}\sum_{k\neq j}^{T_d} C_j C_k] \\ &= \mathbb{E}_c[\sum_i^{T_d} C_i] + \mathbb{E}_c[\sum_j^{T_d}\sum_{k\neq j}^{T_d} C_j C_k] \\ &= \mathbb{E}_c[Cum[C]] + \sum_j^{T_d}\sum_{k\neq j}^{T_d} \mathbb{E}_c[C_j]\,\mathbb{E}_c[C_k] \\ &= \mathbb{E}_c[Cum[C]] + \sum_j^{T_d}\sum_k^{T_d} \mathbb{E}_c[C_j]\,\mathbb{E}_c[C_k] - \sum_m^{T_d}(\mathbb{E}_c[C_m])^2]] \\ &= \mathbb{E}_c[Cum[C]] + \mathbb{E}_c[Cum[C]]^2 - \sum_m^{T_d}(\mathbb{E}_c[C_m])^2\end{aligned} \tag{18}$$

Accordingly, we can compute the variance of $\Delta^T(d, q_t)$ as

$$\begin{aligned}Var_c[\Delta] &= \mathbb{E}_c\left[(\frac{Cum[C] - Cum[\beta]}{Cum[\alpha]} - \mathbb{E}_c[\Delta])^2\right] \\ &= \mathbb{E}_c[(\frac{Cum[C] - Cum[\beta]}{Cum[\alpha]})^2] - \mathbb{E}_c[\Delta]^2 \\ &= \frac{\mathbb{E}_c[Cum[C]] - \sum_m^{T_d}(\mathbb{E}_c[C_m])^2}{(Cum[\alpha])^2} \\ &= \frac{\sum_m^{T_d}(\mathbb{E}_c[C_m]) - \sum_m^{T_d}(\mathbb{E}_c[C_m])^2}{(Cum[\alpha])^2}\end{aligned} \tag{19}$$

Unfortunately, the variance estimated with Eq. (19) is difficult to compute because it involves the computation of expected cumulated clicks and cumulated α (i.e., $\mathbb{E}_c[Cum[C]]$ and $Cum[\alpha]^2$). To further simplify the estimation, we assume that α is restricted in range $[\alpha_{min}, \alpha_{max}]$. Then we have

$$\begin{aligned}Var_c[\Delta] &= \frac{\sum_m^{T_d}(\mathbb{E}_c[C_m]) - \sum_m^{T_d}(\mathbb{E}_c[C_m])^2}{(Cum[\alpha])^2} \\ &< \frac{\sum_m^{T_d}(\mathbb{E}_c[C_m])}{(Cum[\alpha])^2} < \frac{T_d}{(T_d\alpha_{min})^2} \propto \frac{1}{T_d}\end{aligned} \tag{20}$$

We observe that $Var_c[\Delta]$ follows a decay function less than $1/T_d$, which can serve as a good approximation. Empirically, because ranking by $\Delta^T(d, q)$ is more likely to be dominated by old documents when query frequency (T_q) is high, we further encourage exploration based on T_q and approximate $std(\Delta^T(d, q_t))$ with $\sqrt{\log(T_q)/T_d}$.

6 EXPERIMENTS

In this section, we evaluate the effectiveness of proposed techniques with experiments on public LTR datasets. Our goal is to validate whether the proposed unbiased feature projection and uncertain-aware LTR algorithms can effectively construct reliable ranking models by using click data as both features and labels.

6.1 Experimental setup

6.1.1 Datasets. In our experiments, we adopted four publicly available LTR datasets, i.e., MQ2007, MQ2008 [43], MSLR-10K, and MSLR-30K[2]. Table 1 shows the general statistics of each dataset. Queries in each dataset are divided into training, validation, and test partitions. MSLR-10K/MSLR-30K have five-level relevance judgments (from 0 to 4) and MQ2007/MQ2008 have three-level relevance judgments for each query-document pair. The original feature sets of MSLR-10K/MSLR-30K have three behavior features (i.e., feature *134-136*) collected from user behavior data. For reliable evaluation, we remove them in advance. All features in MQ2007/MQ2008 are non-behavior features. Therefore, all datasets only have non-behavior features (i.e., x_{nb}) at the beginning of our experiments.

We note that there are other popular large-scale LTR datasets available to the public, such as Yahoo! Letor Dataset [14] and Istella Dataset [20]. However, these datasets are not applicable to this paper because they contain behavior features collected from previous search logs and do not reveal their identity. The goal of this paper

[2]https://www.microsoft.com/en-us/research/project/mslr/

Table 1: Datasets statistics.

Datasets	# Queries	# Average docs per query	# Unique features
MQ2007	1643	41	46
MQ2008	728	20	46
MSLR-10k	9835	122	133
MSLR-30k	30995	121	133

is to investigate the effect of using clicks as both features and labels in LTR, but the existing behavior features in these datasets could pollute our experiment and make it difficult to see whether the system is affected by the behavior features collected in our experiment or the behavior features already presented in these datasets.

6.1.2 Session Sampling and Simulation. Following previous studies [7, 34, 55, 57], we simulate user interactions to evaluate different LTR models. Specifically, this includes the simulation of search sessions, clicks, and cold-start documents.

Click Simulation. At each time step, we randomly sample a query from the training, validation, or test partition, and then create a ranked list with the current ranking model. The relevance probabilities of each document-query pair are simulated with their relevance label y as $P(R=1|d,q)=\frac{y(d,q)}{y_{max}}$ where y_{max} is 2 or 4 depending on the dataset. Clicks are then simulated with $P(C=1|d,q)$ (computed in Eq. (2)) on top 5 (i.e., $K=5$) of each ranked list with $\alpha=[0.35,0.53,0.55,0.54,0.52]$ and $\beta=[0.65,0.26,0.15,0.11,0.08]$[3]. *Note that we only use the sessions sampled from training partitions to train LTR models.* We sample sessions from validation and test together only to collect user behavior features on the fly.

New Item Simulation. In practice, new documents/items frequently come to the retrieval collection during the serving of LTR systems. To simulate such cold-start scenarios, at the beginning of each experiment, we randomly sample only 5 to 10 documents for each query as their candidate sets and mask all other documents. Then, at each time step t with sampled query q_t, with 50% probability, we randomly sampled one masked document and add it to the candidate set of q_t. Depending on the averaged number of document candidates for each query, this means that we need to simulate roughly 134k, 30k, 2.4M, and 7.6M sessions for MQ2007, MQ2008, MSLR-10K, and MSLR-30K, respectively, in order to add most documents into each query in the simulated experiments.

6.1.3 Baselines. In this paper, we compare the proposed techniques with baselines using different feature settings and exploration strategies for LTR. Specifically, in terms of how to extract and incorporate behavior features, we have four feature settings as

- **Feature-w/o-behav.**: Construct LTR models without behavior features as described in Eq (10).
- **Feature-w-ctr**: Construct LTR models using Eq. (11) with the raw CTR of each document as the behavior features (x_b).
- **Feature-w-Δ**: Construct LTR models using Eq. (11) with the proposed feature $\Delta^T(d|q)$ as the behavior features (x_b).
- **Feature-IE**: Construct LTR models using Eq. (12) where $\Delta^T(d|q)$ is used as independent evidence for ranking.

Also, we tested five ranking exploration algorithms:

- **Top-k**: Create ranked lists with top k documents sorted by LTR model scores (i.e., Eq (13) with $\lambda=0$).
- **Random-k**: Create ranked lists with random k documents from the candidate set D_{q_t} at each time step.
- **EpsilonRank**: Create ranked lists with Eq (13) and $es_t(d,q)$ randomly uniformly sampled from $[0,1]$, similar to [32][4].
- **PDGD**: The state-of-the-art online LTR algorithm [38] that sample ranked lists stochastically with PL distributions.
- **UCBRank**: Our proposed uncertainty-aware LTR algorithm using model uncertainty as $es_t(d,q)$ in Eq (13).

Note that PDGD requires a single scoring function f to conduct stochastic ranking sampling, so it is incompatible with Feature-IE.

6.1.4 Implementation and Model Updates. Compared to updating model parameters online, updating ranking features in LTR systems is relatively easy and time-efficient. To better mimic the periodic training and serving process of LTR systems in practice, we update behavior features more frequently than model parameters in our experiments. Specifically, on each dataset, we update behavior features 100 times and only update model parameters 20 times. For example, on MQ2007 with 20k simulated sessions, this means updating behavior features every 200 sessions while updating model parameters every 1k sessions. For a feature update at time step T, we update behavior features for all query-document pairs with clicks collected before T; For a model update at time step T, we retrain the LTR models only with clicks collected on queries from the training set before T. At time step t, the most recently updated behavior features and ranking model will be used ($T<t$). So there is no risk of future information leakage. Note that this is similar to the practice of LTR systems in real applications where we periodically use the click logs collected before a certain timestamp to compute the behavior features and build training datasets to train/update the service models online.

For LTR models, we implemented the ranking function f using feed-forward neural networks with 2 hidden layers (32 neurons per layer). We implemented the loss function in Eq. (1) with mean square errors and estimated $P(R=1|q,d)$ with $\Delta^T(d|q)$. We set the number of inference trials (i.e., N in Eq. (16)) as 10 for model-based uncertainty estimation. Other hyper-parameters including λ are tuned based on sessions sampled from the validation set. The experimental scripts and implementations used in this paper are available online[5].

6.1.5 Evaluation. We evaluate all models with two standard ranking metrics. The first is the Cumulative NDCG (**c-NDCG**) which measures the overall ranking quality of ranking systems in online learning and serving processes. Following previous studies [57], we use c-NDCG as a discounted accumulation of ranking lists' NDCG at each time step with discounted factor $\gamma=0.995$. The second one is the standard **NDCG** on ranked lists sorted by the final ranking models learned by each algorithm after training. We tested both the situations with (i.e., *warm setting*) or without (i.e., *cold setting*) behavior features collected from click history. For simplicity, we use rank cutoff 5 and compute iDCG in both c-NDCG and NDCG

[3]The value of α and β were chosen based on used studies from Agarwal et al. [2].

[4]EpsilonRank adds random perturbation (i.e., ϵ) to ranking scores directly instead of using it to randomly select items as in [32], but the motivations and effects are similar.

[5]https://github.com/Taosheng-ty/UCBRankSIGIR2022.git

Table 2: The online performance (c-NDCG@5) of each algorithm on the test partitions of each dataset. ∗ and † indicate statistical significance over other models in the same or all feature settings, respectively.

Feature settings	Online Algorithms	Datasets			
		MQ2007	MQ2008	MSLR-10k	MSLR-30k
Feature-w/o-behav.	Top-k	93.14	121.9	103.6	**106.6**
	Random-k	59.96	76.42	59.88	73.55
	EpsilonRank	95.70	119.2	102.9	106.1
	PDGD	86.52	95.93	100.8	98.10
	UCBRank	**97.03***	**128.7***	**104.2**	104.1
Feature-w-ctr	Top-k	113.1	143.7	106.2	105.7
	Random-k	63.58	76.27	61.55	74.52
	EpsilonRank	**151.8***	**163.1***	125.5	133.8
	PDGD	145.3	122.4	117.8	122.0
	UCBRank	149.9	147.7	**142.0***	**134.7**
Feature-w-Δ	Top-k	77.94	109.4	82.03	96.87
	Random-k	60.74	78.42	61.48	74.38
	EpsilonRank	96.51	118.4	100.3	112.4
	PDGD	110.7	161.0	100.4	113.0
	UCBRank	**167.3***	**172.0***	**166.0***†	**163.6***
Feature-IE	Top-k	136.5	149.1	125.5	133.8
	Random-k	58.96	77.18	59.58	71.31
	EpsilonRank	159.1	165.0	138.0	150.2
	UCBRank	**172.0***†	**174.0***†	**159.0***	**167.7***†

with all documents in each dataset. Significant tests are conducted with the Fisher randomization test [51] with $p < 0.05$.

Please note all metrics are computed on the test set using the ground truth relevance labels. There is no risk of label leakage because all models are built with the click data collected in the simulation process on the training set. We do collect clicks on validation and test queries in the simulation process to construct the behavior features for their candidate documents, but those data and features have neither been seen nor used in the training of the LTR models. In this way, our experiment can effectively evaluate LTR systems in both the scenarios where we encounter new queries with no click history on any candidate documents (i.e., the *cold setting*) and the scenarios where the queries are new to the system but some of the candidate documents have behavior features collected from previous click logs (i.e., the *warm setting*).

6.2 Results

We now describe the results of our experiments. Table 2 shows the overall performance of each algorithm in training and online serving. Table 3 shows the performance of the final ranking model with (i.e., warm settings) or without (i.e., cold settings) behavior features collected from previous clicks. As shown in the tables, incorporating behavior features extracted from clicks usually increases the performance of ranking models, e.g., the Feature-w-ctr models are better than their Feature-w/o-behav. versions on c-NDCG@5 for more than 10% in most cases. However, depending on how we extracted and used the behavior features, the improvements varied dramatically among different algorithms.

6.2.1 Does the proposed unbiased feature projection function alleviate biases in behavior features? As discussed in Section 4.2, the unbiased feature projection function can resolve position bias and

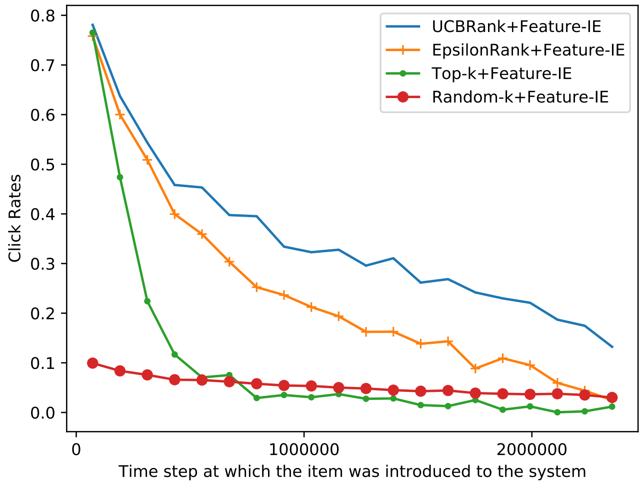

Figure 1: Click rates ($\frac{\#\text{clicks on the item}}{\#\text{sessions with the item as candidate}}$) on *perfect* documents (i.e., label 4) that entered the system at different time steps in the online experiments of MSLR-10k.

trust bias in theory, so we expect models with Feature-w-Δ to outperform models with Feature-w-ctr in all cases. However, according to our experiments, this hypothesis seems to be unreliable. As shown in Table 2&3, except for UCBRank, we haven't observed significant evidence proving that Feature-w-Δ models are superior than Feature-w-ctr models. There could be two explanations. First, the introduction of feature projection with α and β may increase the variance in behavior features, which hurts the robustness of certain algorithms. For example, in contrast to Feature-w-Δ, we observed that Feature-IE models (except for Random-k) always outperform their Feature-w-ctr versions in Table 2. This indicates that, when using together with other features (i.e., Feature-w-Δ), the variance introduced by feature projections may affect complicated ranking models (e.g., neural networks). The second explanation is that the effect of correcting position bias and trust bias has been overwhelmed by the exploitation bias in behavior features. A piece of evidence for this is that, once we alleviated the exploitation bias in behavior feature collections (as discussed later in this section), Feature-w-Δ with UCBRank have achieved consistent improvements over all Feature-w-ctr models. Therefore, exploitation bias is a more significant bias in feature collection that prevents us from directly observing the benefits of unbiased feature projections.

6.2.2 Can our uncertainty-aware LTR framework address the exploitation bias in behavior feature collection? In general, we believe that the exploitation bias in behavior features is solved if all documents, no matter when they are introduced into the system, can be ranked according to their intrinsic relevance correctly. This can be observed in the final performance of LTR models learned by each algorithm in warm settings. As shown in Table 3 (warm settings), after correcting the position bias and trust bias in behavior feature computation, UCBRank have significantly outperformed all other algorithms with the same feature settings in Feature-w-Δ and Feature-IE. Note that this superior performance is achieved in the online simulations where new documents are constantly introduced to each query session. In Figure 1, we plot the click rates

Table 3: The offline performance (NDCG@5) of each algorithm on the test partitions of each dataset. $*$ and $\dagger$ indicate statistical significance over other models in the same or all feature settings, respectively. We tested both situations with (i.e., *warm setting*) or without (i.e., *cold setting*) behavior features collected from the periodic training and serving process.

Feature settings	Online Algorithms	Warm Settings				Cold Settings			
		MQ2007	MQ2008	MSLR-10k	MSLR-30k	MQ2007	MQ2008	MSLR-10k	MSLR-30k
Feature-w/o-behav.	Top-k	0.472	0.610	**0.482**	**0.497**	0.472	0.610	**0.482**	**0.497**
	Random-k	**0.497**	0.607	0.477	0.491	**0.497**	0.607	0.477	0.491
	EpsilonRank	0.486	0.618	0.480	**0.497**	0.486	0.618	0.480	**0.497**
	PDGD	0.493	0.632	0.467	0.464	0.493	0.632	0.467	0.464
	UCBRank	**0.497**	**0.643***	**0.482**	0.496	**0.497**	**0.643***	**0.482**	0.496
Feature-w-ctr	Top-k	0.578	0.717	0.490	0.428	**0.403***	0.505	0.251	0.282
	Random-k	**0.804***	**0.875***	0.676	0.684	0.325	0.445	**0.364***	0.336
	EpsilonRank	0.780	0.839	0.724	0.698	0.364	0.401	0.302	0.324
	PDGD	0.735	0.830	0.565	0.537	0.366	**0.636***	0.342	**0.372***
	UCBRank	0.793	0.747	**0.695***	**0.712***	0.370	0.528	0.331	0.267
Feature-w-Δ	Top-k	0.382	0.541	0.359	0.361	0.381	0.540	0.313	0.226
	Random-k	0.772	0.871	0.609	0.618	0.414	0.552	0.232	0.198
	EpsilonRank	0.505	0.632	0.453	0.454	0.398	0.543	0.265	0.216
	PDGD	0.553	0.836	0.456	0.454	0.379	**0.584***	**0.364***	**0.371***
	UCBRank	**0.864***	**0.885***	**0.815***	**0.819***	**0.415**	0.542	0.239	0.204
Feature-IE	Top-k	0.681	0.758	0.612	0.635	0.461	0.617	0.451	0.485
	Random-k	0.758	0.872	0.587	0.597	**0.515***†	**0.625**	**0.477**	**0.492**
	EpsilonRank	0.818	0.876	0.789	0.755	0.466	0.619	0.475	0.485
	UCBRank	**0.887***†	**0.893***	**0.836***†	**0.825***†	0.496	0.622	0.461	0.489

($\frac{\text{\#clicks on the item}}{\text{\#sessions with the item as candidate}}$) on *perfect* relevant documents (i.e., label 4) that were introduced into the system at different time steps in online experiments. Without addressing the exploitation bias, clicks on old relevant documents (i.e., *perfect* documents introduced at early time steps) dominated the Top-k model, which made new relevant documents (i.e., *perfect* documents introduced at late time steps) received almost no click from users. When we conducted too much exploration (i.e., Random-k), the click rates of relevant documents introduced at different time steps are similarly low because all documents were shown without considering their utility to users. In contrast, with UCBRank, relevant documents received significantly more clicks than those in other algorithms, and the improvement margins are particularly large for new relevant documents. This demonstrates the effectiveness of our uncertainty-aware LTR framework in alleviating exploitation bias and correctly identifying new relevant documents on the fly.

As shown in Figure 2, when we changed the strength of exploration with λ in Eq. (13), we observed that UCBRank with Feature-IE can outperform other uncertainty-oblivious exploration strategies (i.e., EpsilonRank) in all feature settings. This indicates that UCBRank with Feature-IE is a robust algorithm that can effectively extract unbiased behavior features and unbiased ranking models while balancing user experience with ranking exploration.

6.2.3 *Bias in Labels vs. Bias in Features.*

Previous studies on unbiased LTR [2, 34, 39] only focus on biases in training labels (e.g., clicks) while ignoring the effect of biases in LTR features. In this paper, we argue that biases in LTR features cannot be solved by simply resolving biases in labels. In fact, bias in features could

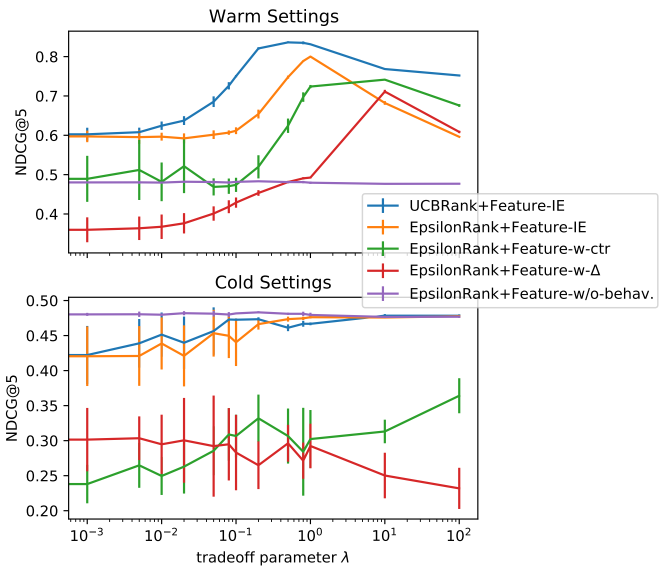

Figure 2: The final test performance (offline) on MSLR-10k with different exploration strength. Greater tradeoff parameter λ (in Eq.(13)) means more explorations.

be a more fundamental problem because addressing feature biases in some cases could alleviate or solve label biases implicitly. For example, the selection bias [40, 41, 59] in training labels is usually considered as an important problem in LTR. However, as shown in Table 3, the final performance of Feature-w/o-behav. algorithms that should severely suffer from the selection bias (i.e., Top-k) haven't shown consistent differences with those that don't suffer from the selection bias (i.e., Random-k). In fact, Top-k even outperforms Random-k and state-of-the-art online LTR algorithm (e.g., PDGD) in Feature-w/o-behav. on MSLR-30k. This contradicts the assumption that the selection bias in training labels should hurt LTR performance. If we investigate the problem from feature perspectives, this is actually not surprising. When ranking features (e.g., the non-behavior features constructed from document statistics) are extracted independently of the biases in training labels (e.g., the selection bias), their distributions on relevant documents would also be independent of the label biases. In this case, as long as clicks are positively correlated to relevance, the selection bias may not affect the final ranking models significantly. For example, when we use BM25 as the only feature in LTR, the selection bias in click labels wouldn't prevent the model to learn that "documents are more likely to be relevant when their BM25 scores are higher".

When there are biases in LTR features, things are much more complicated. On the one hand, any existing biases in labels could be amplified by the biases in features. For instance, in Table 3 (warm settings), the effect of the selection bias is significant in Feature-w-ctr (e.g., Top-k is more than 20% worse than Random-k). On the other hand, biases in features could also significantly hurt the robustness of learning algorithms. When feeding behavior features into neural rankers together with other features (i.e., Feature-w-ctr and Feature-w-Δ), the optimization of model parameters was quickly dominated by behavior features (no matter whether we remove the position/trust biases or not) due to their high correlations to the training labels. As a result, the final models were extremely sensitive to behavior features and performed terribly when behavior features were missing (i.e., the cold settings in Table 3). By treating behavior features as independent evidence (i.e., Feature-IE), such problems can be alleviated and the UCBRank with Feature-IE can significantly outperform other models with behavior features (i.e., warm settings) while keeping a reasonable performance in situations without behavior features (i.e., cold settings).

7 CONCLUSION AND FUTURE WORK

In this paper, we investigate the possibility of using clicks as both labels and features for LTR systems. While behavior features extracted from raw clicks could bring significant improvements to short-term retrieval performance, they could also hurt the long-term effectiveness of LTR systems due to the click noise and exploitation bias in feature collection. To address these problems, we proposed an unbiased feature projection function and an uncertainty-aware LTR algorithm that significantly improve the robustness and effectiveness of LTR models with behavior features.

While the consideration of model uncertainty can help us better explore new items in feature collection, we also observed that existing uncertainty estimation methods for LTR often produce results that suffer from significant variance. As one of the future directions, we plan to explore more on how to conduct effective uncertainty estimation for ranking models and how to use it to guide the training and data collection for LTR.

ACKNOWLEDGEMENTS

This work was supported in part by Amazon Search and in part by the School of Computing, University of Utah. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

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 and Development in Information Retrieval*. 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*. 4–14.

[3] Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 19–26.

[4] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 135–144.

[5] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 385–394.

[6] Qingyao Ai, Jiaxin Mao, Yiqun Liu, and W Bruce Croft. 2018. Unbiased learning to rank: Theory and practice. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2305–2306.

[7] Qingyao Ai, Tao Yang, Huazheng Wang, and Jiaxin Mao. 2021. Unbiased Learning to Rank: Online or Offline? *ACM Transactions on Information Systems (TOIS)* 39, 2 (2021), 1–29.

[8] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3, Nov (2002), 397–422.

[9] ZW Birnbaum and RC McCarty. 1958. A Distribution-Free Upper Confidence Bound for \Pr{Y<X\}, Based on Independent Samples of X and Y. *The Annals of Mathematical Statistics* (1958), 558–562.

[10] Christine L Borgman. 1983. End user behavior on an online information retrieval system: A computer monitoring study. In *Proceedings of the 6th annual international ACM SIGIR conference on Research and development in information retrieval*. 162–176.

[11] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*. 89–96.

[12] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[13] Alexandra Carpentier, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, and Peter Auer. 2011. Upper-confidence-bound algorithms for active learning in multi-armed bandits. In *International Conference on Algorithmic Learning Theory*. Springer, 189–203.

[14] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*. PMLR, 1–24.

[15] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[16] Merlise Clyde and Edward I George. 2004. Model uncertainty. *Statistical science* 19, 1 (2004), 81–94.

[17] Daniel Cohen, Bhaskar Mitra, Oleg Lesota, Navid Rekabsaz, and Carsten Eickhoff. 2021. Not All Relevance Scores are Equal: Efficient Uncertainty and Calibration Modeling for Deep Retrieval Models. *arXiv preprint arXiv:2105.04651* (2021).

[18] 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*. 87–94.

[19] J Shane Culpepper, Charles LA Clarke, and Jimmy Lin. 2016. Dynamic cutoff prediction in multi-stage retrieval systems. In *Proceedings of the 21st Australasian Document Computing Symposium*. 17–24.

[20] 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.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[22] Thomas G Dietterich and Eun Bae Kong. 1995. *Machine learning bias, statistical bias, and statistical variance of decision tree algorithms*. Technical Report. Citeseer.

[23] David Draper. 1995. Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society: Series B (Methodological)* 57, 1 (1995), 45–70.

[24] Allen L Edwards. 1985. *Multiple regression and the analysis of variance and covariance*. WH Freeman/Times Books/Henry Holt & Co.

[25] Nicola Ferro, Claudio Lucchese, Maria Maistro, and Raffaele Perego. 2017. On including the user dynamic in learning to rank. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1041–1044.

[26] Nicola Ferro, Claudio Lucchese, Maria Maistro, and Raffaele Perego. 2020. Boosting learning to rank with user dynamics and continuation methods. *Information Retrieval Journal* 23, 6 (2020), 528–554.

[27] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[28] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. PMLR, 1050–1059.

[29] Aurélien Garivier and Eric Moulines. 2011. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*. Springer, 174–188.

[30] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. 2020. A deep look into neural ranking models for information retrieval. *Information Processing & Management* 57, 6 (2020), 102067.

[31] Parth Gupta, Tommaso Dreossi, Jan Bakus, Yu-Hsiang Lin, and Vamsi Salaka. 2020. *Treating Cold Start in Product Search by Priors*. Association for Computing Machinery, New York, NY, USA, 77–78. https://doi.org/10.1145/3366424.3382705

[32] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2013. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval* 16, 1 (2013), 63–90.

[33] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2017. Accurately interpreting clickthrough data as implicit feedback. In *ACM SIGIR Forum*, Vol. 51. Acm New York, NY, USA, 4–11.

[34] 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.

[35] Yen-Chieh Lien, Daniel Cohen, and W Bruce Croft. 2019. An assumption-free approach to the dynamic truncation of ranked lists. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 79–82.

[36] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[37] Craig Macdonald, Rodrygo LT Santos, and Iadh Ounis. 2012. On the usefulness of query features for learning to rank. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. 2559–2562.

[38] 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*. 1293–1302.

[39] 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*. 489–498.

[40] Harrie Oosterhuis and Maarten de Rijke. 2021. Unifying Online and Counterfactual Learning to Rank: A Novel Counterfactual Estimator that Effectively Utilizes Online Interventions. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 463–471.

[41] Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. 2020. Correcting for selection bias in learning-to-rank systems. In *Proceedings of The Web Conference 2020*. 1863–1873.

[42] Gustavo Penha and Claudia Hauff. 2021. On the Calibration and Uncertainty of Neural Learning to Rank Models for Conversational Search. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 160–170.

[43] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).

[44] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13, 4 (2010), 346–374.

[45] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*. Springer, 232–241.

[46] Haggai Roitman, Shai Erera, and Bar Weiner. 2017. Robust standard deviation estimation for query performance prediction. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. 245–248.

[47] Mohd Sami. [n. d.]. Upper Confidence Bound Algorithm in Reinforcement Learning. *https://www.geeksforgeeks.org/upper-confidence-bound-algorithm-in-reinforcement-learning/*.

[48] Anne Schuth, Robert-Jan Bruintjes, Fritjof Buüttner, Joost van Doorn, Carla Groenland, Harrie Oosterhuis, Cong-Nguyen Tran, Bas Veeling, Jos van der Velde, Roger Wechsler, et al. 2015. Probabilistic multileave for online retrieval evaluation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 955–958.

[49] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave gradient descent for fast online learning to rank. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 457–466.

[50] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. 2012. Predicting query performance by query-drift estimation. *ACM Transactions on Information Systems (TOIS)* 30, 2 (2012), 1–35.

[51] Mark D Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. 623–632.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[53] Anh Tran, Tao Yang, and Qingyao Ai. 2021. ULTRA: An Unbiased Learning To Rank Algorithm Toolbox. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4613–4622.

[54] Arif Usta, Ismail Sengor Altingovde, Rifat Ozcan, and Ozgur Ulusoy. 2021. Learning to Rank for Educational Search Engines. *IEEE Transactions on Learning Technologies* (2021).

[55] 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.

[56] Huazheng Wang, Yiling Jia, and Hongning Wang. 2021. Interactive Information Retrieval with Bandit Feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2658–2661.

[57] Huazheng Wang, Sonwoo Kim, Eric McCord-Snook, Qingyun Wu, and Hongning Wang. 2019. 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*. 835–844.

[58] Huazheng Wang, Ramsey Langley, Sonwoo Kim, Eric McCord-Snook, and Hongning Wang. 2018. Efficient exploration of gradient space for online learning to rank. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 145–154.

[59] 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.

[60] Ryen White. 2013. Beliefs and biases in web search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 3–12.

[61] Ryen W White, Ian Ruthven, and Joemon M Jose. 2002. The use of implicit evidence for relevance feedback in web retrieval. In *European Conference on Information Retrieval*. Springer, 93–109.

[62] Tao Yang, Shikai Fang, Shibo Li, Yulan Wang, and Qingyao Ai. 2020. Analysis of multivariate scoring functions for automatic unbiased learning to rank. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2277–2280.

[63] 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*. 1201–1208.

[64] Jianhan Zhu, Jun Wang, Michael Taylor, and Ingemar J Cox. 2009. Risk-aware information retrieval. In *European Conference on Information Retrieval*. Springer, 17–28.