Addressing Exploitation Bias in Learning to Rank with an Uncertainty-aware Empirical Bayes Approach

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

Cuize Han
Amazon Search
Palo Alto, CA, USA
cuize@amazon.com

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

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

Jeff M. Phillips
University of Utah
Salt Lake City, Utah, USA
jeffp@cs.utah.edu

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

Abstract

Ranking is at the core of many artificial intelligence (AI) applications, including search engines, recommender systems, etc. Modern ranking systems are often constructed with learning-to-rank (LTR) models built from user behavior signals. While previous studies have demonstrated the effectiveness of using user behavior signals (e.g., clicks) as both features and labels of LTR algorithms, we argue that existing LTR algorithms that indiscriminately treat behavior and non-behavior signals in input features could lead to suboptimal performance in practice. Particularly, because user behavior signals often have strong correlations with the ranking objective and they can only be collected on items that have already been shown to users, directly using behavior signals in LTR could create an exploitation bias that hurts the system performance in the long run.

To address the exploitation bias, we propose EBRank, an empirical Bayes based uncertainty-aware ranking algorithm. Specifically, to overcome exploitation bias brought by behaviors features in ranking models, EBRank uses a sole non-behavior feature based prior model to get a prior estimation of relevance. In the dynamic training and serving of ranking systems, EBRank uses the observed user behaviors to update posterior relevance estimation instead of concatenating behaviors as features in ranking models. Besides, EBRank additionally applies an uncertainty-aware exploration strategy to actively explore and collect user behaviors for empirical Bayesian modeling and improve ranking performance. Experiments on three public datasets show that EBRank is effective and practical, and can deliver superior ranking performance compared to state-of-the-art ranking algorithms.

Keywords

Learning to rank, Behavior feature, Exploitation bias

WSDM '22, July 11–15, 2021, online
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Ranking techniques have been extensively studied and used in modern Information Retrieval (IR) systems such as search engine, recommender systems, etc. Among different ranking techniques, learning to rank (LTR), which relies on building machine learning (ML) models to rank items, is one of the most popular ranking frameworks. In particular, industrial LTR systems are usually constructed with user behavior feedback/signals since user behaviours (e.g. click, purchase) are cheap to get and direct indicate results' relevance from the user's perspective [44]. For example, previous studies [6, 21, 25, 44] have shown that, instead of using expensive relevance annotations from experts, effective LTR models can be learned directly from training labels constructed with user clicks. Besides using clicks as training labels, many industrial IR systems have also considered user clicks as features for their LTR models [8]. For example, Agichtein et al. [4] have shown that, by incorporating user clicks as behavior features to ranking systems, the performance of competitive web search ranking algorithms can be improved by as much as 31% relative to the original performance.

However, without proper treatment, LTR with user behaviours could also bring damage to ranking quality in the long term [1, 24, 48]. Specifically, user behavior signals usually have high correlations with the training labels, no matter whether the labels are constructed from expert annotations or from users behaviour. Such high correlation could easily make input features built from user behavior signals, which we refer to as the behavior features, overwhelm other features in training, dominate model outputs, and be over-exploited in inference. Such over-exploitation phenomenon would hurt practical ranking systems when user behavior signals are unevenly collected on different candidate items. For example, user clicks can only be collected on items that have already been shown to users. New items that haven't been shown to users would thus suffer in ranking due to the lack of historical click data, and in turn prevent us from exploring them in future despite of their true relevance, which we referred to as the exploitation bias. Intuitively, the above exploitation bias could be more severe when we use user clicks/behaviors as both labels and features, which is a common practice in LTR [17, 45, 48].

In this paper, we propose to solve the above exploitation bias with uncertainty-aware empirical Bayesian based modelling. Specifically, we consider a general application scenario where a ranking system is built with user behavior signals (e.g. clicks) in both its input and objective function. We show that, without differentiating the treatment of behavior signals and non-behavior signals in input features, existing LTR algorithms could suffer significantly from exploitation bias. Aware of the dynamic changing nature of user behavior signals, we treat user behavior and non-behaviour features without a direct concatenation to avoid non-behavior features being overwhelmed. Specifically, EBRank uses a non-behavior feature based prior model to give a prior relevance estimation. With more behavior data collected from the online serving process of the ranking system, EBRank gradually updates its posterior relevance estimation to give a more accurate relevance estimation. Besides the empirical Bayesian model, we also developed a theoretically principled exploration algorithms that combines the optimization of ranking performance with the minimization of model uncertainty. Experiments on three public datasets show that our final algorithm can effectively overcome exploitation bias and deliver superior ranking performance compared to state-of-the-art ranking algorithms.

2 Related Work

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

Online Learning to Rank. OLTR learns a ranking model by directly interacting with users[19, 42, 51] with some exploration strategy. Since it directly interacts with users, exploration-exploitation tradeoff has to considered to avoid negatively impacting user experience. In recent years, methods [32, 34, 41, 42, 42, 43, 49], belonging to the Bandit learning method group, learn a linear ranking model via examining one or more exploratory gradient directions and update the current ranker. However, those methods usually show inferior performance when using non-linear ranking models [7, 38]. Besides bandit learning, Oosterhuis and de Rijke [26] proposed Pairwise Differentiable Gradient Descent (PDGD), which is a state of the art method relying on neural networks. Recently, Oosterhuis and de Rijke [28] proposed a unifying method which extends counterfactual approaches [3, 5, 21, 47] from the offline setting to work in an online setting, which assumes the knowledge of user behavior pattern. Similar to PDGD, this online counterfactual approach also works for non-linear models.

Behavior feature. Agichtein et al. [4] showed that incorporating user behavior data as features can significantly improve the ranking performance of top results. Qin et al. [30] introduced LTR features including behavior features when releasing several LTR benchmark datasets. Ferro et al. [15, 16] showed performance drops when excluding behavior feature in offline LTR task. Usta et al. [39] showed ranking model trained on features of user behavior outperformed various baselines based on ad-hoc retrieval functions when building a search engine for education purpose. Macdonald et al. [23] conducted experiments which show that employing user behavior features significantly improves upon an effective learned model that does not use any query feature.

Uncertainty in ranking. The concept of uncertainty for ranking was discussed by Zhu et al. [50]. In their work, variance of a probabilistic language model was treated as a risk-based factor to improve ranking performance. Tasks like query performance prediction [31, 35] and query cutoff prediction [12, 22] rely on the uncertainty estimation to improve the ranking performance. In those work, the uncertainty is usually from generative paradigm which treats relevance estimation as the probability of generating a query given a document. Recently, simultaneous works [10, 29] explore the impact of Bayesian and ensemble uncertainty with BERT-based deep ranking model.

Table 1: A summary of notations.

$d, q, D(q)$	For a query q, $D(q)$ is the set of candidates items. $d \in D(q)$ is an item.
e, r, c	All are binary random variables indicating whether an item d is examined ($e = 1$), perceived as relevant ($r = 1$) and clicked ($c = 1$) by a user respectively.
R, p_i, E, π	$R = p(r = 1 \mid d)$, is the probability an item d is perceived as relevant. $p_i = p(e = 1 \mid rnk(d) = i, \pi)$ is the examination probability of item d when it is put in the i^{th} rank in a ranklist π. E is an item's accumulated examination probability (see Eq.6).
k_s, k_c	Users will stop examining items lower than rank k_s due to selection bias (see Eq. 5). k_c is the cutoff prefix to evaluate Cum-NDCG and $k_c \le k_s$.

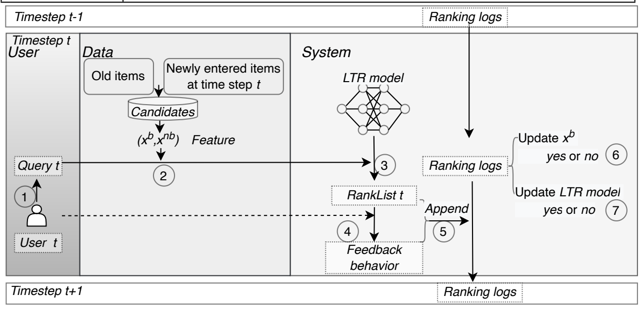

Figure 1: Workflow of ranking services.

3 Problem Formulation

Before introducing the proposed ranking algorithm, we introduce some preliminary knowledge which includes the workflow of ranking services, modelling of users' behavior and utilities to evaluate a ranking system. For better illustration, we give a summary of notations in Table 1.

The Workflow of Ranking Services. In Figure 1, we use web search as an example to introduce the workflow of ranking service in detail, but the method we propose in this paper can also be extended to recommendation or other ranking scenarios. At time step t, a user issues a query q_t. Corresponding to this query, there exist candidate items which include old items and new items introduced at time step t. These items are represented as features which include behaviour features x^b and non-behaviour features x^{nb} (detailed discussion about features are in the following paragraph). Based on the features, a ranking model will predict the relevance of each candidate item and the ranking optimization methods will generate the ranked list by optimizing the ranking objective. Different ranking objectives can be adopted here. For example, the ranking objective can be to put more relevant items on top ranks. After examining the ranked list, the user will provide behavior feedback, such as clicks. The ranklist and user's feedback behavior will be appended to ranking logs. Then we need to decide whether to update the LTR model and behaviour features or not. In practice, such updates are usually conducted periodically since real-time updating ranking systems are often not preferable.

Features. Features can be categorized to two groups. The first group is behavior features, denoted as x^b, are usually derived from user behavior data, which can be click-through-rate, dwelling time, click and etc. x^b are direct and strong indicators of items' relevance from the user's perspective since x^b are collected directly from the user themselves. The second group are non-behavior features, denoted as x^{nb}, generally show items' quality and the degree of matching between item and query. Example non-behavior features can be BM25, query length, tf-idf, page-rank and etc. Compared with behavior features x^b, non-behavior features x^{nb} are usually more stable and static. In this paper, we only focus on one type of user behavior data, i.e., clicks. Extending our work to other types of user behaviors is also straightforward, but we leave them for future studies.

Partial and Biased User Behavior. Although user's feedback behavior are commonly used in LTR, user's feedback behavior usually is biased and partial. Specifically, a user will provide meaningful feedback click ($c = 1$) only when a user actually examines ($e = 1$) the item, i.e.,

$$c = \begin{cases} r, & \text{if} \quad e = 1 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where r indicates if an item d is judged as relevant after being examined. Detailed summary of notations is in Table 1. We formally define the relevance R in this paper as the probability of an item d to be relevant to query q,

$$R(d, q) = P(r = 1|d, q) \tag{2}$$

Following existing works [6, 44], we model users' click behavior with the following,

$$P(c = 1|d, q, \pi) = R(d, q) \times P(e = 1|d, q, \pi) \tag{3}$$

where $P(c = 1|d, q, \pi)$ and $P(e = 1|d, q, \pi)$ are the click probability and examination probability respectively. In this paper, we model users' examination probability with two most important biases, i.e., positional bias and selection bias. The **position bias** [11] assumes that examination probability is decided by items' rank in a ranklist π, i.e.,

$$P(e = 1|d, q, \pi) = P_{rank(\pi,d)} \tag{4}$$

$rank(\pi, d)$ denotes the rank of item d in a ranklist π. The **selection bias** [27, 28] exists when not all of the items are selected to be shown to users or some lists are so long that no user will examine the entire lists. The selection bias is modeled as ,

$$P(e = 1|d, q, \pi) = P_{rank(\pi,d)} = 0 \quad \text{if} \quad rank(\pi, d) > k_s \tag{5}$$

where k_s is the lowest rank that will be examined. We can also define item d's exposure as its accumulated examination probability:

$$E(d, q) = \sum_{i=1}^{t} P_{rank(\pi_i,d)} \tag{6}$$

Exploitation Bias. In Figure 2, we give a toy example to illustrate the exploitation bias. The exploitation bias usually happens because behavior features in LTR models will overwhelm other features since behavior features are strong indicators of relevance and highly correlated with training labels. In this example in Figure 2, due to the overwhelming importance of behavior features, newly introduced item G will be discriminated and ranked low when its

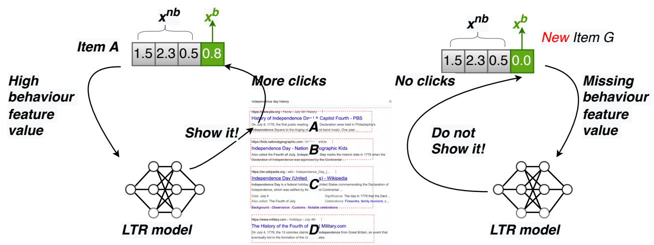

Figure 2: Toy example to show exploitation bias. Old item A and new item E have the same non-behaviour (x^{nb}) features while item G's behaviour features (x^b) is 0 as it has not be shown to users before. Item G will be discriminated since the LTR model over-exploits and heavily relies on x^b.

user behavior information is missing. Being ranked low and failure to collect user feedback will form a vicious cycle where an item will continue to be ranked low will leading to more failure to collect user feedback, and more low ranking, and so on. The vicious cycle will make item G difficult to be discovered as relevant. How to avoid the vicious cycle and discover relevant yet cold new items to improve ranking quality is what we address in this paper.

Ranking utility. In this paper, ranking utility is evaluated by measuring a ranking system's ability to effectively put relevant items on top ranks; such utility is also referred to as *effectiveness*. One widely-used ranking utility measurement is *DCG* [20]. For a ranked list π corresponding to a query q, we define $DCG@k_c$ as

$$DCG@k_c(\pi) = \sum_{i=1}^{k_c} R(\pi[i], q)\lambda_i \tag{7}$$

where $\pi[i]$ indicates the i^{th} ranked item in the ranked list π; $R(\pi[i], q)$ indicates item $\pi[i]$'s relevance to query q; cutoff k_c indicates the top ranks we evaluate; λ_i indicates the weight we put on i^{th} rank. λ_i usually monotonically decreases as rank i increases since top ranks are generally more important. For example, λ_i is sometimes set to $\frac{1}{\log_2(i+1)}$. In this paper, we follow [36] to choose the examining probability p as λ:

$$DCG@k_c(\pi) = \sum_{i=1}^{k_c} R(\pi[i], q) \cdot P_i \tag{8}$$

where P_i is users' examining probability of the i^{th} item in ranked list π_i (refer Eq. 4). Then, we can define the normalized-*DCG* (*NDCG* $\in [0, 1]$) by normalizing $DCG@k_c(\pi)$ with $DCG@k_c(\pi^*)$,

$$NDCG@k_c(\pi) = \frac{DCG@k_c(\pi)}{DCG@k_c(\pi^*)} \tag{9}$$

where π^* is the ideal ranked list constructed by arranging items according to their true relevance. Furthermore, we could define discounted Cumulative NDCG (Cum-NDCG) as,

$$\textit{eff.} = \textit{Cum-NDCG}@k_c = \sum_{\tau=1}^{t} \gamma^{t-\tau} NDCG@k_c(\pi_\tau) \tag{10}$$

where $0 \le \gamma \le 1$ is the discounted factor, t is the current time step. Compared to NDCG, Cum-NDCG can better evaluate ranking effectiveness for online ranking services [33]. Since we mainly consider online ranking scenarios shown in Figure 1, **we use Cum-NDCG as the effectiveness objective and measurement in this work.**

4 Proposed Method

In this section, we propose an uncertainty-aware Empirical Bayes (EB) based learning to rank algorithm, EBRank, which can effectively overcome exploitation bias with EB modelling. First, we propose an uncertainty-aware ranking objective (Section 4.1). To optimize the ranking objective, EBRank relies on two parts, the Bayesian relevance model (Section 4.2) and the marginal-certainty-aware exploration strategy (Section 4.3).

4.1 Uncertainty-Aware ranking optimization

In this section, we propose an uncertainty-aware ranking objective and its optimization.

4.1.1 The uncertainty-aware ranking objective. As shown in Figure 1, at time step t, a user issues a query q, we want to find the optimum ranklist π_t that can maximizes the following uncertainty-aware ranking objective,

$$\max_{\pi_t} \quad Obj(q,t) = \widehat{eff.}(q,t) - \epsilon \widehat{uncert.}(q,t) \tag{11}$$

where $\widehat{eff.}(q,t)$ is the estimated Cumulative-NDCG (see Eq.10) by using the estimated relevance[1] $\hat{R}$ instead of the unavailable true relevance R. $\widehat{uncert.}$ is the uncertainty of relevance estimation.

$$\widehat{uncert.}(q,t) = \sum_{d \in D(q)} Variance(\hat{R}(d,q)) \tag{12}$$

ϵ is the coefficient to balance the two parts. The intuition of the ranking objective in Eq, 11 is that we originally only want to maximize the true *eff.*, but we only have the estimated $\widehat{eff.}$ which is calculated based on estimated $\hat{R}(d,q)$. To effectively optimize *eff.* via optimizing $\widehat{eff.}$, $\widehat{eff.}$ should be accurate estimation of *eff.* which means we need to make $\hat{R}(d,q)$ more certain by minimizing the uncertainty. Please note that here we assume that the true relevance of each (q,d) pair is not available in both the training and serving of ranking systems, thus optimizing the ranking effectiveness with an estimated relevance $\hat{R}$ is the best we can do.

4.1.2 Ranking optimization. At time step t, the optimization of $Obj(q,t)$ is actually equivalent to finding π_t to maximize the marginal objective $\Delta Obj(q,t)$

$$\begin{aligned}
\max_{\pi_t} \quad Obj(q,t) &= \max_{\pi_t} \quad Obj(q,t) - Obj(q,t-1) && (13a)\\
&= \max_{\pi_t} \quad \Delta Obj(q,t) && (13b)\\
&= \max_{\pi_t} \quad \Delta\widehat{eff.}(q,t) - \epsilon\Delta\widehat{uncert.}(q,t) && (13c)\\
&= \max_{\pi_t} \quad NDCG@k(\pi_t) - \epsilon\Delta\widehat{uncert.}(q,t) && (13d)\\
&\doteq \max_{\pi_t} \quad DCG@k(\pi_t) - \epsilon\Delta\widehat{uncert.}(q,t) && (13e)\\
&\approx \sum_{d \in D(q)} \left(\hat{R}(d) + \epsilon MC(d)\right)\Delta E(d) && (13f)\\
&\approx \sum_{d \in D(q)} \left(\hat{R}(d) + \epsilon MC(d)\right)P_{rank(\pi_t,d)} && (13g)
\end{aligned}$$

$$MC(d) = -\frac{\partial \widehat{uncert.}(q,t)}{\partial E(d)} \tag{14}$$

In Eq. 13a, the ranked list π_t at time t will not change $Obj(t-1)$ because π_t cannot change history. So maximizing the ranking objective $Obj(q,t)$ is equivalent to maximizing the marginal ranking objective $\Delta Obj(q,t)$, which can be divided to the marginal effectiveness $\Delta\widehat{eff.}(q,t)$ and the negative marginal uncertainty $\Delta\widehat{uncert.}(q,t)$ in Eq. 13d. The $\Delta\widehat{eff.}(q,t)$ denotes the increment at time t, i.e, $NDCG@k(\pi_t)$ in Eq. 10, and we use $DCG@k(\pi_t)$ to stand for $\Delta\widehat{eff.}(q,t)$ in Eq. 13e by ignoring the normalization in *NDCG* for simplicity. The negative marginal uncertainty $\Delta\widehat{uncert.}(q,t)$ is estimated by using the first order approximation, i.e., $MC(d) \times \Delta E(d)$ in Eq. 13g, where $MC(d)$, the gradient, denotes **M**arginal **C**ertainty, the speed to gain additional certainty. $\Delta E(d)$ is the incremental exposure that item d will get at time t, i.e., $P_{rank(\pi_t,d)}$.

To maximize $Obj(q,t)$ in Eq. 13, according to the Rearrangement Inequality [18], $\forall d \in D_q$, $\left(\hat{R}(d) + \epsilon MC(d)\right)$ and $P_{rank(\pi_t,d)}$ should be in the same descending order. By assuming that $P_{rank(\pi_t,d)}$ monotonically descends as $rank(\pi_t,d)$ goes lower, we would know the optimal π_t should be generated by sorting items according to $\left(\hat{R}(d) + \epsilon MC(d)\right)$ in descending order[2],

$$\pi_t = \arg_{topk}\left(\hat{R}(d) + \epsilon MC(d) | \forall d \in D(q)\right) \tag{15}$$

In the following two sections, we propose how to get relevance estimation $\hat{R}$ (Section 4.2) and marginal certainty MC (Section 4.3).

4.2 Empirical Bayesian relevance model

In this section, we propose an empirical Bayesian (EB) model which gives a posterior relevance estimation $\hat{R}$ based on a sole non-behavior feature x^{nb} based prior model and observations of user clicks. Similar to the concatenation in Fig. 2, the EB model can also take advantage of both behaviour and non-behavior signals to give relevance estimation. However, behavior signals are used as *observation* to update the relevance prior and we use an empirical Bayesian approach to conduct unbiased updates to the ranking systems so that it suffers less from the exploitation bias.

To construct the empirical Bayesian relevance model, we need to model the observation (Section 4.2.1) and model the prior and posterior (Section 4.2.2) to get the posterior estimation. Finally, we introduce how to train and update prior model with observations in Section 4.2.3.

4.2.1 The observation modelling. When there is no position bias and r is directly observable, the probability of observation of r is

$$Prob(r|R) = \log \prod_{i=1}^{N} \prod_{j=1}^{n_i} \left(\left((R_i)^{r_{i,j}} \times (1-R_i)^{(1-r_{i,j})} \right) \right) \tag{16}$$

where N is the number of unique items, n_i is item d_i's number of times being presented to user. $r_{i,j}$ indicates whether item d_i is judged as relevant or not in its jth impression.

However, r is actually unobservable and we can only observe c which is a biased indicator of r according to Eq. 1. Here, we introduce an unbiased counterfactual observation probability $Prob^*(Clicks|R)$,

[1]In this paper, we use the hat notation to denote that it is an estimation

[2]We omit the proof due to page limit.

$$\begin{aligned} Prob^*(Clicks|R) &= \prod_{i=1}^{N}\prod_{j=1}^{n_i}\left(\left((R_i)^{\frac{c_{i,j}}{P_{e_{i,j}}}} \times (1-R_i)^{(1-\frac{c_{i,j}}{P_{e_{i,j}}})}\right)\right) \\ &= \prod_{i=1}^{N}\left(\left(R_i^{C_i} \times (1-R_i)^{n_i-C_i}\right)\right) \quad (17) \\ C_i &= \sum_{j=1}^{n_i}\frac{c_{i,j}}{P_{e_{i,j}}} \end{aligned}$$

where $c_{i,j}$ indicates whether item d_i is clicked or not in its j^{th} presentation, and C_i is After taking log, $Prob^*(Clicks|R)$ is an unbiased estimation of $Prob(r|R)$ in the sense that,

$$\begin{aligned} &\mathbb{E}_e[\log Prob^*(Clicks|R)] \\ &= \sum_{i=1}^{N}\sum_{j=1}^{n_i}\left(\frac{\mathbb{E}_e[c_{i,j}]}{P_{e_{i,j}}}\log(R_i) + (1-\frac{\mathbb{E}_e[c_{i,j}]}{P_{e_{i,j}}})\log(1-R_i)\right) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{n_i}\left(r_{i,j}\log(R_i) + (1-r_{i,j})\log(1-R_i)\right) \quad (18) \\ &= \log Prob(r|R) \end{aligned}$$

where $\mathbb{E}_e[c_{i,j}] = P_{e_{i,j}} r_{i,j}$ using Eq. 1 and Eq. 3.

4.2.2 The prior & posterior distribution. With modeling of the probability of observations, now we turn to model the prior and posterior distributions of relevance. By observing that the formulation $Prob^*(Clicks|R)$ is similar to a binomial distribution, we choose binomial distribution's conjugate prior, the Beta distribution, as the prior distribution to estimate relevance, i.e., the density of R_i's prior estimation is

$$\begin{aligned} p(R_i|\theta) &= \frac{R_i^{\alpha_i-1}(1-R_i)^{\beta_i-1}}{B(\alpha_i,\beta_i)} \quad (19) \\ (\alpha_i,\beta_i) &= f_\theta(x_i^{nb}) \end{aligned}$$

where B denotes the beta function, (α_i, β_i) are outputs of trainable model f_θ with only x_i^{nb} as input. Combining the likelihood of observation and prior distribution, we can get the joint probability,

$$\begin{aligned} Prob(Clicks, R|\theta) &= Prob^*(Clicks|R)\prod_{i=1}^{N}p(R_i|\theta) \\ &= \prod_{i=1}^{N}\frac{R_i^{C_i+\alpha_i-1}(1-R_i)^{n_i-C_i+\beta_i-1}}{B(\alpha_i,\beta_i)} \quad (20) \end{aligned}$$

From above joint probability, we know that the posterior distribution of R_i also follows a beta distribution,

$$\begin{aligned} Prob(R_i|Clicks,\theta) &\propto R_i^{C_i+\alpha_i-1}(1-R_i)^{n_i-C_i+\beta_i-1} \\ &= Beta(C_i+\alpha_i, n_i-C_i+\beta_i) \quad (21) \end{aligned}$$

And we use the mean of the posterior distribution as the posterior relevance estimation $\widehat{R}$ to rank items (see Eq. 15),

$$\widehat{R}(d_i) = \mathbb{E}[R_i|Data,\theta] = \frac{C_i+\alpha_i}{n_i+\alpha_i+\beta_i} \quad (22)$$

In above equation, as long as $n_i > 0$, $\frac{C_i}{n_i}$ itself is an unbiased estimation of relevance R_i,

$$\mathbb{E}_c[\frac{C_i}{n_i}] = \frac{\sum_{j=1}^{n_i}\mathbb{E}_c[c_{i,j}]/P_{e_{i,j}}}{n_i} = \frac{\sum_{j=1}^{n_i}R_i}{n_i} = R_i \quad (23)$$

$\frac{\alpha_i}{\alpha_i+\beta_i}$ is the mean of R_i's prior distribution(Eq. 19), which tries to predict relevance R_i (see Sec. 4.2.4). The posterior's mean in Eq. 22 is actually a blending between $\frac{C_i}{n_i}$ and $\frac{\alpha_i}{\alpha_i+\beta_i}$. The posterior relevance estimation $\widehat{R}$ can overcome exploitation bias by nature since it will rely more on $\frac{\alpha_i}{\alpha_i+\beta_i}$ when n_i and C_i are are small i.e., new items. And $\widehat{R}$ gradually relies more on $\frac{C_i}{n_i}$ when n_i and C_i increase and start to dominate i.e., more user behaviors are observed.

4.2.3 Update prior distribution with observations. The values (α_i, β_i), from prior model $f_\theta(x_i^{nb})$ (see Eq. 19), are a critical component in the posterior relevance estimation $\widehat{R}$. To get the optimal θ, we perform a maximal likelihood estimation by maximizing the marginal likelihood of the observed data [14] i.e., $Prob(Clicks|\theta)$.

$$\begin{aligned} Prob(Clicks|\theta) &= \int\cdots\int_R Prob(Clicks,R|\theta)dR_1dR_2... \\ &= \prod_{i=1}^{N}\frac{\int_0^1 R_i^{C_i+\alpha_i-1}(1-R_i)^{n_i-C_i+\beta_i-1}dR_i}{B(\alpha_i,\beta_i)} \quad (24) \\ &= \prod_{i=1}^{N}\frac{B(C_i+\alpha_i, n_i-C_i+\beta_i)}{B(\alpha_i,\beta_i)} \end{aligned}$$

Maximize the above probability is also equivalent to minimizing the negative log likelihood,

$$Loss = \sum_{i=1}^{N}\log B(\alpha_i,\beta_i) - \log B(C_i+\alpha_i, n_i-C_i+\beta_i) \quad (25)$$

We optimize this loss periodically after collecting some new clicks and then update the prior model f_θ.

4.2.4 Convergence Analysis. To investigate what is the optimal (α_i, β_i) during training, we first take the derivative of the loss function,

$$\frac{\partial Loss}{\partial \alpha_i} = \psi(\alpha_i) - \psi(\alpha_i+\beta_i) - \left(\psi(C_i+\alpha_i) - \psi(n_i+\alpha_i+\beta_i)\right) \quad (26)$$

where ψ is the Digamma function. Usually, by setting, $\frac{\partial Loss}{\partial \alpha_i} = 0$, we could know the optimal (α_i^*, β_i^*). However, from the best of our knowledge, it is difficult to directly get (α_i^*, β_i^*) since Digamma function is a special function. Since $\psi(x) \approx \log(x)$ with error $O(\frac{1}{x})$[2], we use log function to substitute ψ in Eq. 26, set $\frac{\partial Loss}{\partial \alpha_i} = 0$, and we can get,

$$\frac{\alpha_i^*}{\alpha_i^*+\beta_i^*} - \frac{C_i+\alpha_i^*}{n_i+\alpha_i^*+\beta_i^*} = 0 \quad (27)$$

It is straightforward to see that prior estimation $\frac{\alpha_i^*}{\alpha_i^*+\beta_i^*}$ should be $\frac{C_i}{n_i}$ which is an unbiased estimation of relevance R_i considering Eq. 23. However, how accurate the learnt $\frac{\alpha_i}{\alpha_i+\beta_i}$ can approximate R_i depends on the capability of the non-behavior features and the prior model function f_θ. Theoretically, with the perfect features and ranking function, the optimal solution of Eq. 27 is guaranteed.

Because the focus of this paper is not to the design of non-behavior features or ranking functions, we ignore the discussion of them here.

Actually, the proposed EB model provides a more reasonable way than concatenation to blend behaviour and non-behaviour signals, which is not limited to the loss in Eq. 25. For example, our method can be empirically extended to get a ranking score $\tilde{R} = \tilde{f}_\theta(x^{nb})$ by training $\sum_i loss(\tilde{f}_\theta(x_i^{nb}), \frac{C_i}{n_i})$, where the loss can be conventional ranking loss. Similar to Eq. 22, the final relevance score could be $\frac{C_i + w\tilde{R}}{n_i + w}$, where w is a hyper-parameter to denote how we trust non-behaviour features and $\tilde{R}$ has been scaled to $(0, 1)$ in some way.

4.3 Estimation of Marginal Certainty.

After introducing the relevance estimation $\widehat{R}$, the next step is to get the $MC(d)$ used in Eq. 15 to reduce uncertainty. To get $MC(d)$, as indicated in Eq. 14, we first need to get the variance of $\widehat{R}$,

$$
\begin{aligned}
Variance[\widehat{R}] &= \frac{\sum_{j=1}^{n_i} \frac{Var[c_{i,j}]}{p_{i,j}^2}}{(n_i + \alpha_i + \beta_i)^2} = \frac{\sum_{j=1}^{n_i} \frac{1}{p_{i,j}^2} \left(\mathbb{E}[c_{i,j}^2] - \mathbb{E}^2[c_{i,j}] \right)}{(n_i + \alpha_i + \beta_i)^2} \\
&= \frac{\sum_{j=1}^{n_i} \frac{1}{p_{i,j}^2} \left(\mathbb{E}[c_{i,j}] - \mathbb{E}^2[c_{i,j}]) \right)}{(n_i + \alpha_i + \beta_i)^2} = \frac{\sum_{j=1}^{n_i} R_i / P_{e_{i,j}} - R_i^2}{(n_i + \alpha_i + \beta_i)^2} \\
&< \frac{R_i \sum_{j=1}^{n_i} 1/P_{min}}{(n_i + \alpha_i + \beta_i)^2} < \frac{1/P_{min} \times R_i}{n_i + \alpha_i + \beta_i} \doteq \frac{R_i}{E_i + \alpha_i + \beta_i}
\end{aligned}
$$

where $c_{i,j}^2 = c_{i,j}$ since $c_{i,j}$ is a binary random variable, $n_i \geq E_i$ according to Eq. 6. And we think there usually exists a smallest examining probability P_{min} in consideration and we ignore the constant factor, i.e, $1/P_{min}$, in final formulation. With the variance, according to Eq. 12 and Eq. 13, we derive $MC(d)$ as,

$$
MC(d) = \frac{\widehat{R}_i}{(E_i + \alpha_i + \beta_i)^2} \tag{28}
$$

where we use $\widehat{R}_i$ to substitute relevance R_i since R_i is unavailable.

4.4 The final algorithm: EBRank

With Empirical Bayesian relevance modelling and marginal-certainty-aware exploration, we finally propose the **E**mpirical **B**ayesian **Rank**ing (EBRank) in Algorithm 1. In Algorithm 1, we noticed some initial ranking logs usually already exist before using EBRank, and EBRank's prior model can be trained based on the initial logs if the initial logs already exist. After EBRank is deployed, users issue queries and new candidates will be constantly introduced to each query's candidates set. To serve users, EBRank will will first compute $\widehat{R}(q_t, d)$ and $MC(q_t, d)$ and then generate ranklist according to Eq. 15. After user's interaction with the ranklist, the current session's user, query, ranklist and clicks will appended to ranking logs. Finally, we update and retrain the prior model periodically.

5 Experiments

In this section, we evaluate the effectiveness of proposed method with semi-simulation experiments on public LTR datasets.

Algorithm 1: EBRank

```
$t \leftarrow 0$;
$logs = [u_t, q_t, \pi_t, c_t]\ \forall t \in history$;
Initialize the trainable parameters $\theta$ for prior model $f_\theta$;
Train prior model $f_\theta$ with loss in Eq. 25;
while True do
    $t \leftarrow t + 1$;
    User $u_t$ issues a query $q_t$;
    if New candidates introduced then
        $D_{q_t}$.append(new candidates)
    Get $\widehat{R}(q_t, d)$ and $MC(q_t, d)\ \forall d \in D_{q_t}$ via Eq. 22&28;
    Output ranklist $\pi_t$ via Eq. 15;
    Present $\pi_t$ to User $u_t$ and collect user's behaviour $c_t$;
    $logs.append([u_t, q_t, \pi_t, c_t])$;
    if Prior-model-update then
        Train prior model $f_\theta$ with loss in Eq. 25
```

Table 2: Datasets statistics. For each dataset, the table below shows the number of queries, the average number of docs for each query, number of features, the relevance annotation y's range, and the id of BM25 in the features.

Datasets	# Queries	#AverDcos	# Features	y	BM25
MQ2007	1643	41	46	$0-2$	25th
MSLR-10k	9835	122	133	$0-4$	110th
MSLR-30k	30995	121	133	$0-4$	110th

5.1 Experimental setup

5.1.1 Dataset. In the semi-simulation experiments, we will adopt three publicly available LTR datasets, i.e., MQ2007, MSLR-10K, and MSLR-30K, with data statistics shown in Table 2. Queries in each dataset are divided into three partitions, training, validation, and test. For each query-document pair of each datasets, relevance judgment y is provided. The original feature sets of MSLR-10K/MSLR-30K have three behavior features (i.e., feature 134, feature 135, feature 136) collected from user behaviors. For reliable evaluation, we remove them in advance. All features in MQ2007 are non-behavior features. We note that there exist other popular large-scale LTR datasets available to the public, such as Yahoo! Letor Dataset [8] and Istella Dataset [13]. However, these datasets are not applicable to this paper because they contain behavior features but do not reveal their identity. Therefore, all datasets we use only have non-behavior features (i.e.,x^{nb}) at the beginning of our experiments.

5.1.2 Simulation of Search Sessions, Click and Cold-start. Following previous studies [7, 21, 40, 42], we simulate user interactions to evaluate different LTR algorithms. At each time step, we randomly sample a query from the training, validation, or test partition, and then create a ranked list according to ranking algorithms. Following the methodology proposed by Chapelle et al. [9], we convert the relevance judgement to relevance probability according to $P(r = 1|d, q, \pi) = 0.1 + (1 - 0.1)\frac{2^y - 1}{2^{y_{max}} - 1}$, where y_{max} is 2 or 4 depending on the dataset. Besides relevance probability, the examination probability $P_{rank(\pi,d)}$ are simulated with $P_{rank(\pi,d)} = \frac{1}{\log_2(rank(\pi,d)+1)}$. With $P(r = 1|d, q, \pi)$ and $P_{rank(\pi,d)}$, according to Equation 3, clicks can sampled. We simulate clicks on top 5 items, i.e., $k_s = 5$ in Equation 5. User clicks will be simulated and collected for all three partitions. And we use the sessions sampled from training partitions to train LTR models and sessions sampled from validation partitions to do validation. LTR models are evaluated and compared based on test partitions.

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 the initial candidate sets and mask all other documents. For each query, we simulated 20 sessions according to BM25 scores to collect initial user behaviours for the initial candidate sets. The BM25 scores are already provided in the original datasets' features (see Table 2). After we finished the initial user behaviours collection, at each time step t (begin with 0), with probability η ($\eta = 1.0$ by default), we randomly sample one masked document and add it to the candidate set of q_t. Depending on the averaged number of document candidates and η, the number of sessions we will simulate for each dataset is

$$\#Session = \frac{\#Queries \times (\#AverDcos - 5)}{\eta} \tag{29}$$

where $\#Queries$ and $\#AverDcos$ are indicated in Table 2.

5.1.3 Baselines. To evaluate our proposed methods, we have the following ranking algorithms to compare,

BM25: For reference, we include a trivial baseline which only use BM25 to rank items.

CFTopK: Train a ranking model with Counterfactual L2 loss [24, 46, 48] and create ranked lists with items sorted by the model scores during ranking service.

CFRandomK: Train a ranking model the same way as CFTopK, but randomly create ranked lists with items during ranking service.

CFEpsilon: Train a ranking model the same way as CFTopK. Uniformly sample an exploration from $[0, 1]$ and add to each item's score from the model[48]. Create ranked lists with items sorted by the score after addition during ranking service.

DBGD[49]: A learning to rank method which samples one variation of a ranking model and compares them using online evaluation during ranking service.

MGD[34]: Similar to DBGD but sample multiple variations.

PDGD[26]: A learning to rank method which decides gradient via pairwise comparison during ranking service..

Prior[17]: It train a behaviour feature prediction model which only uses x^{nb}. The missing behavior feature's default value is not 0 but predicted by the behaviour feature prediction model.

UCBRank[48]: An online ranking algorithm, which relies on a hard switch strategy to combine behaviour and non-behaviour signals. UCBRank chooses to use behaviour signals when they are available, otherwise use non-behaviour signals.

EBRank: Our method shown in Algorithm 1.

We compare those methods with two feature settings. The first setting that is we only use non-behavior features x^{nb}, referred to as **W/o-Behav**. The second feature setting is that we use both behavior signals and non-behaviour features. Feature setting is referred to as **W/-behav(Concate)** when behaviour derived features and non-behaviour features are concatenated together. **W/-behav(Non-Concate)** means behaviour signals and non-behaviour features are combined using a non-concatenation way, such as the EB modelling used by EBRank. Method BM25 only has W/o-Behav results since it only uses non-behaviour feature BM25 to rank items. CFTopK, CFRandomK, CFEpsilon, DBGD, MGD, and PDGD have ranking performance with both W/o-Behav and W/-behav(Concate) feature settings, depending on whether behaviour features are used or not. We follow Yang et al. [48] to use relevance's unbiased estimator $x^b = \frac{C_i}{n_i}$ (in Eq. 23) as the behaviour feature, with default value as 0 when $n_i = 0$. For UCBRank and EBRank, they only have W/-behav(Non-Concate) results since they have their own designed non-concatenation way to combine behaviour signals and non-behaviour features.

5.1.4 Implementation. Linear model are used all methods except BM25. And we retrain and update ranking model parameters periodically for 20 times during total simulation sessions (see Eq. 29). Compared to updating model parameters online, updating ranking logs including user behaviours in LTR systems are relatively easy and time-efficient, and we update them after each session. When we train the prior model according to loss in Eq. 25, we only train $\alpha = f_\theta(x^{nb})$ and fix $\beta = 5$ for simplicity, which works well across all experiments.

5.1.5 Evaluation. We evaluate all ranking methods with two standard ranking metrics. The first is the Cumulative NDCG (**Cum-NDCG**) (see Eq.10) with the discounted factor $\gamma = 0.995$ to evaluate online performance of presented ranklists. The second one is the standard **NDCG** (Eq.9) where each queries' ranked list is generated by sorting scores from the final ranking models (excluding the exploration strategy part of each algorithm). The NDCG is used to evaluate the offline performance of learnt ranking model. We tested NDCG in two the situations with (i.e., **warm-NDCG**) or without (i.e., **cold-NDCG**) behavior features collected from click history. In this way, our experiment can effectively evaluate LTR systems in scenarios where we encounter new queries with no user behaviour history on any candidate documents (i.e., the *cold-NDCG*) and the scenarios where some of the candidate documents have behavior features collected from previous click logs while new candidates do not have (i.e., the *warm-NDCG*). For simplicity, we use rank cutoff as 5 and compute iDCG in both Cum-NDCG and NDCG with all documents in each dataset. Significant tests are conducted with the Fisher randomization test [37] with $p < 0.05$. All the evaluations are based on five independent trials and reported on test partition only.

5.2 Result

In this section, we will describe the results of our experiments.

5.2.1 How does our method compared with baselines? In Table 3, EBRank significantly outperforms all other methods+ feature setting combinations on Warm-NDCG and Cum-NDCG while its Cold-NDCG are among the best. The discussion in the following sections will give more insights of EBRank's supremacy.

5.2.2 Will historical user behaviours help ranking algorithms to get better ranking quality? As shown in Table 3, not all algorithms can benefit from incorporating behaviour signals. Particularly, in Table 3, we indeeed see that both W/-behav(Concate) and W/-behav(Non-Concate) feature settings help to boost most ranking algorithms to have better Warm-NDCG and Cum-NDCG. However,

Table 3: The ranking performance with standard deviation in the parentheses. $*$ indicates the best performance. Cold-NDCG and Warm-NDCG are the same for each algorithms with the W/o-Behav feature setting since behaviour signals is not used in this setting.

Feature settings	Online Algorithms	MQ2007			MSLR-10k			MSLR-30k		
		Cold-NDCG	Warm-NDCG	Cum-NDCG	Cold-NDCG	Warm-NDCG	Cum-NDCG	Cold-NDCG	Warm-NDCG	Cum-NDCG
	BM25	$0.474_{(0.001)}$	$0.474_{(0.001)}$	$94.38_{(1.414)}$	$0.449_{(0.001)}$	$0.449_{(0.001)}$	$90.39_{(1.783)}$	$0.451_{(0.000)}$	$0.451_{(0.000)}$	$89.98_{(1.225)}$
	DBGD	$0.557_{(0.015)}$	$0.557_{(0.015)}$	$110.6_{(4.180)}$	$0.488_{(0.007)}$	$0.488_{(0.007)}$	$98.66_{(3.586)}$	$0.498_{(0.005)}$	$0.498_{(0.005)}$	$99.40_{(3.006)}$
	MGD	$0.562_{(0.013)}$	$0.562_{(0.013)}$	$110.9_{(5.557)}$	$0.473_{(0.012)}$	$0.473_{(0.012)}$	$95.72_{(1.878)}$	$0.502_{(0.009)}$	$0.502_{(0.009)}$	$101.3_{(1.789)}$
W/o-Behav	PDGD	$\mathbf{0.599}^{*}_{(0.003)}$	$0.599_{(0.003)}$	$115.0_{(2.760)}$	$\mathbf{0.525}^{*}_{(0.012)}$	$0.525_{(0.012)}$	$105.2_{(3.620)}$	$\mathbf{0.525}^{*}_{(0.006)}$	$0.525_{(0.006)}$	$106.0_{(2.764)}$
	CFTopK	$0.591_{(0.007)}$	$0.591_{(0.007)}$	$116.7_{(4.515)}$	$0.510_{(0.022)}$	$0.510_{(0.022)}$	$102.9_{(4.725)}$	$0.506_{(0.006)}$	$0.506_{(0.006)}$	$101.6_{(2.421)}$
	CFRandomK	$0.589_{(0.005)}$	$0.589_{(0.005)}$	$87.69_{(1.389)}$	$0.509_{(0.012)}$	$0.509_{(0.012)}$	$79.59_{(1.132)}$	$0.514_{(0.009)}$	$0.514_{(0.009)}$	$78.63_{(1.362)}$
	CFEpsilon	$0.589_{(0.009)}$	$0.589_{(0.009)}$	$94.39_{(0.963)}$	$0.510_{(0.003)}$	$0.510_{(0.003)}$	$84.75_{(0.981)}$	$0.518_{(0.005)}$	$0.518_{(0.005)}$	$83.88_{(1.357)}$
	DBGD	$0.514_{(0.033)}$	$0.729_{(0.020)}$	$144.7_{(5.301)}$	$0.451_{(0.022)}$	$0.571_{(0.021)}$	$114.3_{(5.027)}$	$0.462_{(0.028)}$	$0.607_{(0.017)}$	$118.8_{(9.075)}$
	MGD	$0.523_{(0.043)}$	$0.725_{(0.026)}$	$142.1_{(5.155)}$	$0.461_{(0.025)}$	$0.558_{(0.012)}$	$109.0_{(1.721)}$	$0.444_{(0.024)}$	$0.595_{(0.027)}$	$122.5_{(4.491)}$
W/-Behav (Concate)	PDGD	$0.574_{(0.010)}$	$0.745_{(0.009)}$	$147.6_{(3.229)}$	$0.466_{(0.037)}$	$0.591_{(0.003)}$	$117.9_{(1.394)}$	$0.480_{(0.021)}$	$0.584_{(0.006)}$	$116.4_{(3.477)}$
	CFTopK	$0.385_{(0.044)}$	$0.579_{(0.009)}$	$113.5_{(3.655)}$	$0.369_{(0.029)}$	$0.489_{(0.005)}$	$97.53_{(2.369)}$	$0.366_{(0.033)}$	$0.491_{(0.003)}$	$97.65_{(2.174)}$
	CFRandomK	$0.377_{(0.018)}$	$0.771_{(0.007)}$	$87.11_{(0.736)}$	$0.403_{(0.044)}$	$0.596_{(0.002)}$	$79.82_{(0.936)}$	$0.404_{(0.029)}$	$0.603_{(0.002)}$	$78.91_{(1.922)}$
	CFEpsilon	$0.387_{(0.031)}$	$0.789_{(0.009)}$	$143.8_{(1.245)}$	$0.355_{(0.016)}$	$0.683_{(0.002)}$	$116.8_{(0.746)}$	$0.354_{(0.010)}$	$0.686_{(0.001)}$	$116.8_{(2.161)}$
	Prior	$0.597_{(0.004)}$	$0.791_{(0.005)}$	$158.7_{(1.377)}$	$0.507_{(0.006)}$	$0.554_{(0.009)}$	$110.3_{(2.458)}$	$0.503_{(0.006)}$	$0.557_{(0.005)}$	$111.4_{(1.535)}$
W/-Behav (non-Concate)	UCBRank	$0.593_{(0.005)}$	$0.799_{(0.005)}$	$158.9_{(1.995)}$	$0.514_{(0.007)}$	$0.703_{(0.007)}$	$140.1_{(1.875)}$	$0.509_{(0.010)}$	$0.703_{(0.004)}$	$140.5_{(1.959)}$
	EBRank(ours)	$0.596_{(0.007)}$	$\mathbf{0.849}^{*}_{(0.012)}$	$\mathbf{171.3}^{*}_{(1.630)}$	$0.513_{(0.002)}$	$\mathbf{0.762}^{*}_{(0.003)}$	$\mathbf{151.6}^{*}_{(0.791)}$	$0.513_{(0.002)}$	$\mathbf{0.762}^{*}_{(0.001)}$	$\mathbf{152.0}^{*}_{(2.128)}$

such boosting on Warm-NDCG and Cum-NDCG does not apply to all ranking algorithms and there exist two exceptions. The first one is a trivial exception regarding CFRandomK. CFRandomK always randomly ranks items and shows them to users so its online performance, i.e., Cum-NDCG, can not be boosted. The second one is a **non-trivial exception** regarding CFTopK. Incorporating user behaviours even makes CFTopK degenerate on Warm-NDCG and Cum-NDCG for all three datasets. Besides CFTopK's degeneration on Warm-NDCG and Cum-NDCG, W/-behav(Concate) feature setting even cause all ranking algorithms experience a significant drop in Cold-NDCG, when compared to the W/o-behav feature setting. Compared to other algorithms, UCBRank and our algorithm EBRank can benefit from behaviour signals to have better Warm-NDCG and Cum-NDCG while avoiding drop in Cold-NDCG.

5.2.3 How does ranking algorithms suffer from exploitation bias? In the above section, we found that there exists some exception, i.e., degeneration of ranking quality, when using the W/-behav(Concate) feature setting. In this section, we dig into the learnt models to investigate the reason of the degeneration. To investigate the learnt models, we output behavior and non-behavior features' exploitation ratio for each algorithm in Table 4, where j^{th} feature's exploitation ratio is defined as

$$Ratio_j = \frac{w_j}{\sum_i |w_i|} \tag{30}$$

w_j is the learnt j^{th} weight of the learnt linear model. In Table 4, behaviour feature's exploitation ratio is significantly greater than non-behavior features' ratio for all ranking algorithms, which makes behaviour features overwhelm other non-behaviour features, dominate the output and discriminates cold-start items. The overwhelming effect will cause the exploitation bias mentioned in Fig. 2. Although the analysis is based on linear model, we believe non-linear models will make exploitation bias even more severe since non-linear models are even better at over-fitting the behaviour features.

The exploitation bias causes the ranking quality degeneration mentioned in the last Section (see Sec. 5.2.2). Firstly, CFTopK's degeneration in Warm-NDCG and Cum-NDCG is because it fully trusts the exploitation biased ranking model without any exploration, which makes new items of high quality discriminated and hard to be discovered. Secondly, the drop of Cold-NDCG under the W/-behav(Concate) feature setting is because the exploitation biased ranking model can not give effective ranklist when only non-behaviour features are available.

Table 4: Features' exploitation ratio (Eq. 30) in the learnt linear model. x^b Ratio is behavior features' exploitation ratio. Max x^{nb} Ratio is the maximum exploitation ratio of non-Behaviour features. Exploitation ratios for UCBRank, EBRank and BM25 are NA since they do not contain x^b in the linear model.

Algorithms	x^b Ratio	Max x^{nb} Ratio
CFEpsilon	$0.604_{(0.098)}$	$0.071_{(0.038)}$
CFRandomK	$0.498_{(0.067)}$	$0.103_{(0.026)}$
CFTopK	$0.591_{(0.087)}$	$0.098_{(0.033)}$
DBGD	$0.109_{(0.021)}$	$0.063_{(0.019)}$
MGD	$0.110_{(0.026)}$	$0.062_{(0.018)}$
PDGD	$0.623_{(0.037)}$	$0.036_{(0.004)}$
UCBRank	NA	NA
EBRank	NA	NA
BM25	NA	NA

5.2.4 Is EBRank robust to exploitation bias? In the last section, we reveal the existence of exploitation bias when algorithms use the W/o-Behav feature setting. Compared to those algorithms, EBRank is more robust to exploitation bias since it has comparable ranking performance on Cold-NDCG as algorithms using W/o-behav and has superior performance on Warm-NDCG and Cum-NDCG. In other words, EBRank can take advantage of historical user behavior signals but also robust to user behavior missing. Besides, EBRank also significantly outperforms UCBRank in Cum-NDCG and Warm-NDCG, which means EBRank's BE based merging strategy of behaviour and non-behavior signals is more effective than the hard switch used by UCBRank.

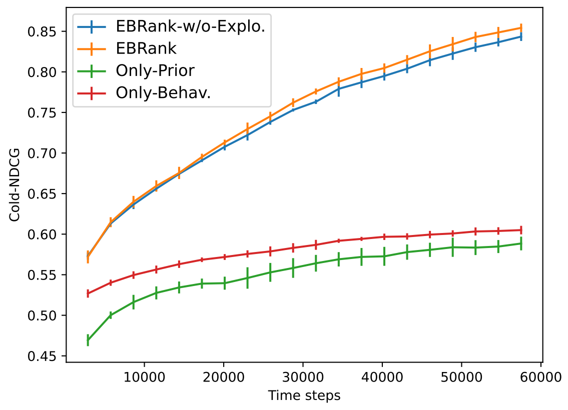

Figure 3: Ablation Study of EBRank. EBRank-w/o-Explo. means excluding $MC(d)$ in Eq. 15 when generaing ranklists. Only-Prior means only using the prior model part $\frac{\alpha_i}{\alpha_i+\beta_i}$ of $\hat{R}$ (in Eq. 22) to rank items. Only-Behav. means only using the behaviour part $\frac{C_i}{E_i}$ of $\hat{R}$ (in Eq. 22) to rank items.

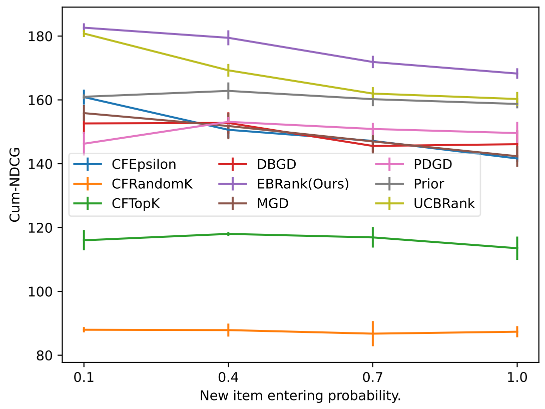

Figure 4: Ranking performance with different entering probability η in simulation (see Eq. 29) on MQ2007. We only consider using user behaviour situations here.

5.2.5 Ablation Study. In this section, we do an ablation study to see whether each part of our EBRank is needed. As shown in Figure 3, EBRank significantly outperforms if we only use Behavior part or use prior model part to rank. Also, from the ablation study, we would know Bayesian modelling can already help a ranking model to reach good ranking quality and be robust to exploitation bias. The marginal-certainty-aware exploration additionally helps to discover relevant items, which helps to boost ranking performance in the long term.

5.2.6 EBRank's robustness to entering probability. To investigate the item entering speed (η) 's influence on the ranking performance, we showed the experimental results with different η in Figure 4. As shown in Figure 4, EBRank consistently significantly outperforms all other algorithms under different item entering speed.

6 CONCLUSION

In this work, we propose an empirical Bayes based uncertainty-aware ranking algorithm EBRank with the aim to overcome the exploitation bias in LTR. With empirical Bayes modelling and a novel exploration strategy, EBRank aims to effectively overcome the exploitation bias and improving ranking quality. Extensive experiments on public datasets demonstrate that EBRank can achieve significantly better ranking performance than state-of-the-art LTR ranking algorithms. In the future, we plan to extend current work to further consider other types of user behaviours beyond user clicks.

References

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 42–46.

[2] Milton Abramowitz and Irene A Stegun. 1964. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Vol. 55. US Government printing office.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[9] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*. 621–630.

[10] 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).

[11] 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.

[12] 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.

[13] 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.

[14] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39, 1 (1977), 1–22.

[15] 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.

[16] 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.

[17] Parth Gupta, Tommaso Dreossi, Jan Bakus, Yu-Hsiang Lin, and Vamsi Salaka. 2020. Treating Cold Start in Product Search by Priors. In *Companion Proceedings of the Web Conference 2020*. 77–78.

[18] Godfrey Harold Hardy, John Edensor Littlewood, George Pólya, György Pólya, et al. 1952. *Inequalities*. Cambridge university press.

[19] 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*. 183–192.

[20] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[21] 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.

[22] 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.

[23] 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.

[24] Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. 2020. Controlling Fairness and Bias in Dynamic Learning-to-Rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 429–438. https://doi.org/10.1145/3397271.3401100

[25] Harrie Oosterhuis. 2022. Doubly-Robust Estimation for Unbiased Learning-to-Rank from Position-Biased Click Feedback. *arXiv preprint arXiv:2203.17118* (2022).

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] Anne Schuth, Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2013. Lerot: An online learning to rank framework. In *Proceedings of the 2013 workshop on Living labs for information retrieval evaluation*. 23–26.

[34] 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.

[35] 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.

[36] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2219–2228.

[37] 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.

[38] 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.

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] Haode Yang, Parth Gupta, Roberto Fernández Galán, Dan Bu, and Dongmei Jia. 2021. Seasonal Relevance in E-Commerce Search. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4293–4301.

[46] Tao Yang and Qingyao Ai. 2021. Maximizing marginal fairness for dynamic learning to rank. In *Proceedings of the Web Conference 2021*. 137–145.

[47] 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.

[48] Tao Yang, Chen Luo, Hanqing Lu, Parth Gupta, Bing Yin, and Qingyao Ai. 2022. Can clicks be both labels and features? Unbiased Behavior Feature Collection and Uncertainty-aware Learning to Rank. (2022).

[49] 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.

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

[51] Masrour Zoghi, Tomas Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2017. Online learning to rank in stochastic click models. In *International Conference on Machine Learning*. PMLR, 4199–4208.