Safe Exploration for Optimizing Contextual Bandits

ROLF JAGERMAN, University of Amsterdam, The Netherlands
ILYA MARKOV, University of Amsterdam, The Netherlands
MAARTEN DE RIJKE, University of Amsterdam, The Netherlands

Contextual bandit problems are a natural fit for many information retrieval tasks, such as learning to rank, text classification, recommendation, etc. However, existing learning methods for contextual bandit problems have one of two drawbacks: they either do not explore the space of all possible document rankings (i.e., actions) and, thus, may miss the optimal ranking, or they present suboptimal rankings to a user and, thus, may harm the user experience. We introduce a new learning method for contextual bandit problems, Safe Exploration Algorithm (SEA), which overcomes the above drawbacks. SEA starts by using a baseline (or production) ranking system (i.e., policy), which does not harm the user experience and, thus, is safe to execute, but has suboptimal performance and, thus, needs to be improved. Then SEA uses counterfactual learning to learn a new policy based on the behavior of the baseline policy. SEA also uses high-confidence off-policy evaluation to estimate the performance of the newly learned policy. Once the performance of the newly learned policy is at least as good as the performance of the baseline policy, SEA starts using the new policy to execute new actions, allowing it to actively explore favorable regions of the action space. This way, SEA never performs worse than the baseline policy and, thus, does not harm the user experience, while still exploring the action space and, thus, being able to find an optimal policy. Our experiments using text classification and document retrieval confirm the above by comparing SEA (and a boundless variant called BSEA) to online and offline learning methods for contextual bandit problems.

CCS Concepts: • **Information systems** → **Learning to rank**.

Additional Key Words and Phrases: Learning to rank, Exploration, Counterfactual learning

ACM Reference Format:
Rolf Jagerman, Ilya Markov, and Maarten de Rijke. 2020. Safe Exploration for Optimizing Contextual Bandits. *ACM Transactions on Information Systems* 1, 1, Article 1 (January 2020), 23 pages. https://doi.org/10.1145/3385670

1 INTRODUCTION

A *multi-armed bandit* problem is a problem in which a limited number of resources must be allocated between alternative choices so as to maximize their expected gain. In *contextual* bandit problems, when making a choice, a representation of the context is available to inform the decision. Contextual bandit problems are a natural framework to capture a range of Information Retrieval (IR) tasks [1, 10, 14]. Example tasks to which contextual bandits have been applied include news recommendation [31], text classification [43], ad placement [28], and online learning to rank [13]. For example, in online Learning to Rank (LTR) (see Figure 1):

Authors' addresses: Rolf Jagerman, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam, The Netherlands, rolf.jagerman@uva.nl; Ilya Markov, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam, The Netherlands, i.markov@uva.nl; Maarten de Rijke, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam, The Netherlands, derijke@uva.nl.

1046-8188/2020/1-ART1 $15.00
https://doi.org/10.1145/3385670

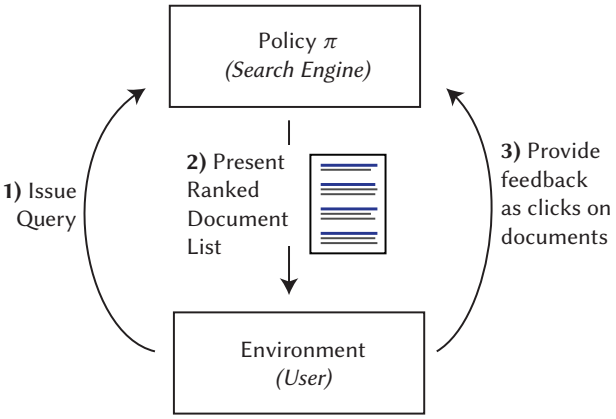

Fig. 1. Online learning to rank viewed as a contextual bandit problem. A user issues a query, the search engine responds with a ranked list of documents, and the user provides implicit feedback in the form of clicks on documents.

(1) a user issues a query,
(2) a search engine presents a ranked document list, and
(3) clicks on the documents are recorded as feedback.

The interactive nature of this problem makes it an ideal application for the contextual bandit framework. The LTR model is a decision-making *policy* π, the ranked document lists it produces are *actions*, the user is the *context* or *environment*, and clicks are the *reward* signal.

There are two major classes of algorithms to maximize the reward of a contextual bandit policy π. The first are online algorithms, which optimize a policy while it is being executed [2, 24, 31, 47]. The second are offline algorithms, which optimize a policy based on data that was collected using an existing logging policy, often called π_0 [39, 43].

Online learning methods for contextual bandits are widely studied and there are many known algorithms to solve this problem. E.g., popular algorithms include ϵ-greedy, LinUCB [31] and Thompson Sampling [2]. Despite their attractive properties, the adoption of online learning methods for contextual bandits in production systems has been limited. Especially in the early stages of learning, online algorithms may perform actions that are suboptimal and, thus, hurt the user experience. E.g., in online LTR it is risky to present suboptimal rankings of documents [34, 46].

Offline learning from logged feedback [11, 12, 39, 43] has been proposed as a solution to this problem. Using an existing and already deployed logging policy, one only takes actions from that logging policy to collect bandit feedback. Using this collected data, a new policy is then learned offline in an unbiased manner. This learning process is safe in the sense that the new policy is not executed and, thus, the user experience is not affected. The drawback of offline learning, however, is that it relies on the deployed logging policy, which never changes by design. Due to this, there may be areas in the action space left unexplored, i.e., bandit feedback may be missing for those areas. In IR terms, potentially high quality document rankings might never be presented to users and the corresponding user interactions will never be observed.

In this paper, we propose a method that performs *exploration* of the action space in contextual bandit problems, but does this exploration *safely*, that is, without hurting the user experience. In our proposed solution, called Safe Exploration Algorithm (SEA), we use an already deployed logging policy which is safe, static but suboptimal, as a *warm-start* for learning a new policy. As

soon as SEA is confident that the performance of the new policy does not fall below that of the logging policy, it starts executing actions from the new policy, thus, exploring the action space. In the context of LTR, this means that SEA is capable of exploring and presenting new rankings to users, gathering feedback for rankings that might otherwise have never been presented. This enables SEA to trade off various strengths and weaknesses of both *online* and *offline* learning.

We note that periodic deployment of a policy, after a successfull evaluation on held-out data, is a common practice in industry. In our experiments we include a boundless version of SEA, called BSEA, which represents such a deployment pipeline. What SEA adds over and above a standard deployment procedure is: (1) it comes with a formal proof of safety, i.e., with high probability the performance of SEA will be at least as good as that of a baseline policy (Section 3.4); and (2) we introduce a computationally efficient manner for computing high-confidence off-policy bounds (Section 3.3).

The research questions we address in this paper are:

(RQ1) Is SEA safe? I.e., does it always perform at least as good as a baseline policy?
(RQ2) Does SEA provide a better user experience during training than offline or online learning methods? I.e., does it accumulate a higher reward during training?
(RQ3) Does SEA, which explores the action space, learn a more effective policy compared to offline learning methods, which do not perform exploration?

Our key technical contributions in this paper are: (1) we introduce SEA and show it to be safe: its performance never falls below that of a baseline policy; (2) we show that SEA improves the user experience: it provides higher cumulative reward during training than both offline and online methods; and (3) we show that a policy learned with SEA, which is capable of exploring new actions, outperforms that of offline methods, which are incapable of exploration.

2 BACKGROUND

2.1 Contextual bandits

In online LTR we try to optimize the parameters of a ranking model such that it places relevant items at the top of the ranked list. We can view the ranking model as a decision-making policy. When a new query arrives, the policy takes an action: it displays one possible ranked list to the user. The user then decides to click on documents shown in the ranked list, thus providing a reward signal. More formally, we consider the following contextual bandit framework. At each round t:

(1) The environment announces an m-dimensional *context vector* $\boldsymbol{x}_t \in \mathcal{X}$. In IR terms, this would be a user (environment) issuing a query (context vector).
(2) A policy π samples an action $a_t \in \mathcal{A}$, one of n possible actions, conditioned on $\boldsymbol{x}_t$: $a_t \sim \pi(\cdot \mid \boldsymbol{x}_t)$. In IR terms, this would be a ranking model (policy) producing a ranked list (action).
(3) The environment announces only the reward r_{t,a_t} for the chosen action a_t, and not for other possible actions that the policy could have taken. We assume that $r \in [0, 1]$. In IR terms, this reward signal could be clicks on documents.

This learning setup is inherently different from supervised learning, where rewards for all possible actions are known. We only observe rewards for actions that the policy has taken. The setting is referred to as *partial-label problem* [25], *associative bandit problem* [40] or *associative reinforcement learning* [24]. In the context of LTR, this means that we do not know the optimal ranked list but can only observe a reward signal for rankings that our policy chooses to display.

There are two major classes of algorithms for learning a contextual bandit policy. The first are online algorithms that optimize the policy while it is being executed [2, 24, 31, 47]. Algorithms of this class explore the space of possible actions but may harm the user experience, because suboptimal actions could be executed. The second are offline algorithms, which optimize a policy

based on data that was collected using an existing logging policy [39, 43]. Algorithms of this class are safe as the newly learned policy is not executed and, thus, there is no risk of hurting the user experience. However, offline algorithms are incapable of exploring the action space, which may harm the performance of the learned model. In this work we build on the second class of algorithms; see below.

2.2 Offline learning from logged bandit feedback

Offline learning algorithms collect bandit feedback using an existing logging policy, which we call a *baseline policy* π_b (also referred to as π_0 in the literature [22, 43, 44]). This feedback is collected in the following form, where at time t we have:

$$\mathcal{D}_t = \left\{\left(\boldsymbol{x}_i, a_i, r_{i,a_i}, p_i\right)\right\}_{i=1}^{t}, \tag{1}$$

where $\boldsymbol{x}_i$ is the observed context vector, a_i is the action taken by the baseline policy, $r_{i,a_i} \in [0, 1]$ is the reward given by the environment, and p_i is the propensity score for action a_i. The propensity score is the probability of the baseline policy taking the logged action, that is, $p_i = \pi_b(a_i \mid \boldsymbol{x}_i)$ [43].

To learn a new policy π_w from the collected bandit feedback, the following maximization problem has to be solved [22]:

$$\widehat{\pi}_w = \max_{\pi_w} \frac{1}{t} \sum_{i=1}^{t} \frac{r_{i,a_i}}{p_i} \pi_w(a_i \mid \boldsymbol{x}_i). \tag{2}$$

This optimization problem is solved via Stochastic Gradient Descent (SGD): if a new tuple $(\boldsymbol{x}_i, a_i, r_{i,a_i}, p_i)$ is observed, we weigh the derivative of $\pi_w(a_i \mid \boldsymbol{x}_i)$ by $\frac{r_{i,a_i}}{p_i}$ and update the weights of π_w using SGD. We refer to this offline learning method as Inverse Propensity Scoring (IPS) [22].

2.3 High-confidence off-policy evaluation

In an offline learning setting, a newly learned policy π_w is never executed, so its performance cannot be measured directly. Instead, we *estimate* its performance using the collected bandit feedback $\mathcal{D}_t$ at time t. The estimated reward of a policy π can be written as [32]:

$$\hat{R}(\pi, \mathcal{D}_t) = \frac{1}{t} \sum_{i=1}^{t} \hat{R}_i = \frac{1}{t} \sum_{i=1}^{t} \frac{r_{i,a_i}}{p_i} \pi(a_i \mid \boldsymbol{x}_i). \tag{3}$$

This estimate suffers from high variance, especially when the propensity scores are small. To resolve this issue, Thomas et al. [45] propose several high-confidence off-policy estimators. These estimators first calculate a confidence interval around the policy's performance and then use the lower bound on this performance for off-policy evaluation. The high-confidence off-policy estimator that we use is based on the Maurer & Pontill empirical Bernstein inequality. The confidence bound can be written as [45]:

$$CB(\pi, \mathcal{D}_t) = \frac{7b \ln\left(\frac{2}{\delta}\right)}{3(t-1)} + \frac{1}{t}\sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{t-1} \sum_{i,j=1}^{t} (\hat{R}_i - \hat{R}_j)^2}, \tag{4}$$

where $(1-\delta) \in [0, 1]$ is the confidence level and b is an upper bound on $\hat{R}$. When both r and p are bounded, b can be calculated exactly: $b = \frac{\max r}{\min p}$. The lower confidence bound on the performance of a policy at time t, $LCB(\pi, \mathcal{D}_t)$, can be computed as:

$$LCB(\pi, \mathcal{D}_t) = \frac{1}{t} \sum_{i=1}^{t} \hat{R}_i - CB(\pi, \mathcal{D}_t). \tag{5}$$

Similarly, we compute the *upper* confidence bound on the performance of a policy as follows:

$$UCB(\pi, \mathcal{D}_t) = \frac{1}{t}\sum_{i=1}^{t}\hat{R}_i + CB(\pi, \mathcal{D}_t). \tag{6}$$

The estimator described in this section provides an effective way to compute a lower and upper bound on the performance of a policy without executing it.

In this paper we build on the estimators designed by Thomas et al. [45], but provide the following contributions: First, we develop SEA, an algorithm for automatic safe exploration by deploying new models (Section 3.2); second, we provide an efficient way to compute the high-confidence bounds (Section 3.3); and, third, we formally prove that SEA is indeed safe (Section 3.4).

3 SAFE EXPLORATION ALGORITHM (SEA)

We define the notion of *safety* as part of the learning process (Section 3.1), walk through the Safe Exploration Algorithm (SEA) (Section 3.2), address the problem of efficient off-policy evaluation (Section 3.3), formally prove the safety of SEA (Section 3.4), and then analyse SEA (Section 3.5).

3.1 Safety

We use the definition of *safety* from so-called conservative methods [26, 49]. These methods make a key assumption: there exists a baseline policy π_b whose actions can always be executed without risk. This assumption is very reasonable in practice. In most industrial settings there exists a production system that we can consider to be the baseline policy. A learning algorithm is considered *safe* if its performance at any round t is at least as good as the baseline policy. Thus, safety is a concept that is always measured relative to a baseline.

More formally, let us first consider the notion of regret at round t during training. We define the *regret* at time t as the cumulative difference in reward obtained by executing actions from our policy π compared to a perfect policy π^*, one that always chooses the action with maximum reward. For notational simplicity, we denote the action that a policy chooses in response to a context vector $\boldsymbol{x}$ as $\pi(\boldsymbol{x})$:

$$Regret_t(\pi) = \sum_{i=1}^{t}(r_{i,\pi^*(\boldsymbol{x}_i)} - r_{i,\pi(\boldsymbol{x}_i)}). \tag{7}$$

A policy π is considered *safe* if its regret is always at most as large as that of the safe baseline policy π_b, i.e., for every $t = 1, \ldots, T$:

$$Regret_t(\pi) \leq Regret_t(\pi_b) \tag{8}$$

or

$$\sum_{i=1}^{t}(r_{i,\pi^*(\boldsymbol{x}_i)} - r_{i,\pi(\boldsymbol{x}_i)}) \leq \sum_{i=1}^{t}(r_{i,\pi^*(\boldsymbol{x}_i)} - r_{i,\pi_b(\boldsymbol{x}_i)}) \tag{9}$$

so

$$\frac{1}{t}\sum_{i=1}^{t} r_{i,\pi(\boldsymbol{x}_i)} \geq \frac{1}{t}\sum_{i=1}^{t} r_{i,\pi_b(\boldsymbol{x}_i)}. \tag{10}$$

In other words, at every time t, we want the average reward of our policy π to be at least as large as the average reward of the baseline policy π_b. In practice, we cannot observe the average reward of a policy without executing it. This is problematic because we cannot know if a policy is safe until we execute it. Fortunately, the off-policy estimators described in Section 2.3 provide a way to *estimate* the performance of a policy without executing it.

3.2 A walkthrough of SEA

The Safe Exploration Algorithm (SEA) learns a new policy π_w *offline* from the output of a baseline policy π_b using offline learning techniques described in Section 2.2. At each iteration t, SEA estimates the performance of the newly learned policy π_{w_t} and the currently deployed policy π_d using the high-confidence off-policy evaluators described in Section 2.3. The new policy is only deployed online when its estimated performance is above that of the existing deployed policy. This allows the newly learned policy to take over and start exploring. This only happens once SEA is confident enough that the new policy's performance will be satisfactory and safe to execute.

Algorithm 1 Safe Exploration Algorithm (SEA)

1: π_b *// Baseline policy (current production system)*
2: $\pi_{w_0} \leftarrow \pi_b$ *// Policy to be learned (initialized with baseline weights)*
3: $\pi_d \leftarrow \pi_b$ *// The deployed policy that is executing actions*
4: $\mathcal{D}_0 \leftarrow \{\}$
5: **for** $t = 1 \dots T$ **do**
6: $\quad \boldsymbol{x}_t \leftarrow$ contextual feature vector at time t
7: $\quad a_t \sim \pi_d(\cdot \mid \boldsymbol{x}_t)$
8: $\quad p_t \leftarrow \pi_d(a_t \mid \boldsymbol{x}_t)$
9: $\quad$ Play a_t and observe reward r_{t,a_t}
10: $\quad \mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t, a_t, r_{t,a_t}, p_t)\}$
11: $\quad w_t \leftarrow w_{t-1} + \eta \frac{r_{t,a_t}}{p_t} \nabla_w \pi_{w_{t-1}}(a_t \mid \boldsymbol{x}_t)$ *// Update weights via gradient ascent*
12: $\quad$ Compute $LCB(\pi_{w_t}, \mathcal{D}_t)$ using Equation 5
13: $\quad$ Compute $UCB(\pi_d, \mathcal{D}_t)$ using Equation 6
14: $\quad$ **if** $LCB(\pi_{w_t}, \mathcal{D}_t) \geq UCB(\pi_d, \mathcal{D}_t)$ **then**
15: $\quad\quad \pi_d \leftarrow \pi_{w_t}$ *// Deploy new policy only when it is safe to do so*
16: $\quad$ **end if**
17: **end for**

The pseudocode for the Safe Exploration Algorithm (SEA) is provided in Algorithm 1. SEA starts from a baseline policy π_b (Line 1) and a new policy that we wish to optimize, π_{w_0} (Line 2). At the start, SEA deploys the policy π_d, which is initialized to the baseline policy (Line 3). At every iteration t, a context vector $\boldsymbol{x}_t$ is observed (Line 6). SEA draws an action from the deployed policy and computes its propensity score (Lines 7–8). SEA executes the chosen action a_t and observes the reward r_t (Line 9). The policy π_w is then updated via SGD (Line 11). We then update the confidence bounds on the estimated performance for the new policy π_{w_t} and the deployed policy π_d (Lines 12–13). When the estimated lower bound performance of π_{w_t} is better than the estimated upper bound performance of π_d (Line 14), we deploy π_{w_t}, such that future actions are executed by the newly learned policy instead of the previous deployed policy π_d.

3.3 Efficient policy evaluation

To implement SEA we use the off-policy estimator described in Section 2.3. Such an implementation will be computationally expensive, because at each round T we have to compute a sum over all t data points collected so far. E.g., in Equation 5, we need to compute the mean

$$\frac{1}{t} \sum_{i=1}^{t} \hat{R}_i$$

and the variance

$$\frac{1}{t}\sum_{i,j=1}^{t}\left(\hat{R}_i - \hat{R}_j\right)^2,$$

at every round t. Therefore, the complexity of applying Equation 5 or 6 would be $O(T)$ and the complexity of Algorithm 1 would be $O(T^2)$.

Recall that $\mathcal{D}_t = \{(\mathbf{x}_i, a_i, r_{i,a_i}, p_i)\}_{i=1}^t$ is the collected log data and that there are $|\mathcal{A}|$ possible actions and $|\mathcal{X}|$ possible contexts. We can then compute the mean $\frac{1}{t}\sum_{i=1}^t \hat{R}_i$ as follows (similar results hold for computing the variance term):

$$\frac{1}{t}\sum_{i=1}^{t}\frac{r_{i,a_i}}{p_i}\pi(a_i \mid \boldsymbol{x}_i) = \frac{1}{t}\sum_{a\in\mathcal{A}}\sum_{\boldsymbol{x}\in\mathcal{X}}\sum_{i=1}^{t}\mathbf{1}\left[a_i = a \wedge \boldsymbol{x}_i = \boldsymbol{x}\right]\frac{r_{i,a_i}}{p_i}\pi(a \mid \boldsymbol{x}) \tag{11}$$

$$= \frac{1}{t}\sum_{a\in\mathcal{A}}\sum_{\boldsymbol{x}\in\mathcal{X}}\pi(a \mid \boldsymbol{x})\sum_{i=1}^{t}\mathbf{1}\left[a_i = a \wedge \boldsymbol{x}_i = \boldsymbol{x}\right]\frac{r_{i,a_i}}{p_i}, \tag{12}$$

where $\mathbf{1}[\cdot]$ is the indicator function.

The key insight here is that the inner sum, $\sum_{t=1}^{T}\mathbf{1}\left[a_t = a \wedge \boldsymbol{x}_t = \boldsymbol{x}\right]\frac{r_{t,a_t}}{p_t}$ can be efficiently computed online during data collection and is independent of the policy π_w that is being evaluated. Specifically, we can create a zero-initialized matrix $W \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{X}|}$ where, each time a new data point $(\boldsymbol{x}_i, a_i, r_{i,a_i}, p_i)$ is logged, we update an entry as follows:

$$W_{a_i,\boldsymbol{x}_i} \leftarrow W_{a_i,\boldsymbol{x}_i} + \frac{r_{i,a_i}}{p_i}.$$

This results in the following method for computing the mean:

$$\frac{1}{t}\sum_{a\in\mathcal{A}}\sum_{x\in\mathcal{X}}\pi(a \mid x)W_{a,x} = \frac{1}{t}\sum_{(a,x):W_{a,x}\neq 0}\pi(a \mid x)W_{a,x}, \tag{13}$$

which can be orders of magnitude faster to compute when $T \gg |\mathcal{A}|\cdot|\mathcal{X}|$. This is not unreasonable in practice, as the size of interaction logs is usually much larger than the number of users and items. In addition, we only need to compute the above sum for a and x for which $W_{a,x} \neq 0$. This enables the use of sparse data structures that can speed up computation even further.

3.4 Proof of safety

SEA is an *online* learning method that we claim to be *safe*. To show that SEA is actually safe, we bound the probability that a suboptimal policy, one whose expected reward is lower than the expected reward of the deployed policy, will be deployed during the duration of the algorithm.

Theorem 3.1. *At any time t, with probability at least $1-2\delta$, SEA will not deploy a suboptimal policy.*

Proof. For notational simplicity we write the expected reward of a policy π as:

$$R_t(\pi) = \mathbb{E}_{a\sim\pi}\left[\frac{1}{t}\sum_{i=1}^{t} r_{i,a}\right]. \tag{14}$$

First, from Algorithm 1 we know that we only deploy π_{w_t} when $LCB(\pi_{w_t}) > UCB(\pi_d)$. Suppose that our confidence bounds do not fail. In this case, it is impossible to deploy a suboptimal policy, since

$$R_t(\pi_{w_t}) \geq LCB(\pi_{w_t}) > UCB(\pi_d) \geq R_t(\pi_d) \tag{15}$$

Consequently, a suboptimal policy can only be deployed when either the confidence bound estimate on our newly learned policy π_{w_t} or our deployed policy π_d fails.

The high-confidence off-policy estimators (see Equations 5 and 6) provide a lower and upper bound on the estimated performance. According to Theorem 1 in [45], these confidence bounds hold with probability at least $1 - \delta$. Conversely, this means that the confidence bounds may fail with probability at most δ:

$$P\left(R_t(\pi) \notin [LCB(\pi), UCB(\pi)]\right) \leq \delta, \tag{16}$$

and, as a result, the probability that either the confidence bound on π_{w_t} or π_d will fail is at most 2δ:

$$P\left(R_t(\pi_{w_t}) \notin \left[LCB(\pi_{w_t}), UCB(\pi_{w_t})\right] \vee R_t(\pi_d) \notin [LCB(\pi_d), UCB(\pi_d)]\right) \leq 2\delta. \tag{17}$$

Therefore, with probability at least $1 - 2\delta$, we will not deploy a suboptimal policy. □

In our experiments (see Section 4) we set $\delta = 0.05$, which means that the algorithm is safe with probability at least $1 - 2\delta = 0.90$ at any time t. We observe that this lower bound is fairly loose because we do not observe a single suboptimal deployment across many repetitions and possible deployment moments.

3.5 Analysis

Conservative Linear Contextual Bandits (CLUCB) [26] is, to our knowledge, the only other online learning method for contextual bandits that is also safe. Unfortunately, CLUCB comes with two limitations:

(1) CLUCB does not scale beyond toy problems because it constructs confidence sets around parameters, which requires solving a constraint optimization problem every time an action has to be chosen which makes it infeasible for realistic IR problems; and
(2) CLUCB can only be applied to linear models.

Our method addresses both limitations. First, SEA scales to large and complex datasets, because it merely needs to compute the lower confidence bound and perform a gradient update step, both of which can be done highly efficiently. Second, SEA can easily be adapted to non-linear models such as gradient-boosted decision trees and neural networks as it is based on gradient descent.

SEA, being an online learning method, is capable of exploration. In contrast, *offline* methods do not explore, they merely observe what a baseline policy is doing. If this baseline policy does not explore well, offline learning techniques, whilst still able to learn, are less effective [43]. And even if the baseline policy is highly explorative, what truly matters is how well this policy explores the regions with favorable losses [36]. SEA solves this problem as it starts exploring actions using the newly learned policy as soon as it is safe to do so. Since the policy learned by SEA has a higher estimated performance than the baseline policy, the actions that it eventually takes are likely to be actions with high reward.

We note that the safety that SEA guarantees does come at a cost: SEA cannot guarantee that it will explore new actions beyond the initial deployed policy. In contrast, purely *online* methods can explore without any restrictions and as a result they may end up learning a better policy than SEA. Put differently, SEA provides a trade-off between safety and exploration: with a lower δ, SEA will be more safe, but gives up some amount of exploration. Vice versa, a higher δ allows SEA to explore more aggressively while giving up some level of safety.

4 EXPERIMENTAL SETUP

To answer our research questions, we consider two tasks: *text classification* and *document ranking*. We consider these two complementary tasks as they differ in the size of the action space, which is

Table 1. Datasets used for the text classification task.

Dataset	Classes	Features	Train size	Test size
USPS [16]	10	256	7,291	2,007
20 Newsgroups [27]	20	62,060	15,935	3,993
RCV1 [29]	53	47,236	15,564	518,571

small in the case of text classification (the number of classes) but large in the case of document ranking (the number of possible ranked lists). In both cases we turn a supervised learning problem into a bandit problem.

4.1 Text classification task

For *text classification* we use a dataset $\mathcal{D} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$, where $\boldsymbol{x}_t \in \mathbb{R}^m$ is a feature representation of the object we wish to classify and $y_t \in \{0, \ldots, n\}$ is the correct label for that object. E.g., in text classification $\boldsymbol{x}_t$ is a bag-of-words representation of a document we wish to classify and y_t the correct label for that document. We are interested in the contextual bandit formulation for multi-class classification, i.e., where the correct label y_t is not known, but we can observe a reward signal for a chosen label a_t.

We follow the methodology of Beygelzimer and Langford [3] to transform a supervised learning problem into a contextual bandit problem:

(1) The environment presents the policy with the feature vector $\boldsymbol{x}_t$ (e.g., the bag-of-words representation of a text document).
(2) The policy chooses a_t, one of n possible labels, as its prediction.
(3) The environment returns a reward of 1 if the chosen label was correct and a reward of 0 if it was incorrect.

Methods that use counterfactual learning (IPS and SEA), require a stochastic baseline policy and the corresponding propensity scores, i.e., the probability that the baseline policy chose the selected label a_t. To this end we define our data-collection policy using the ϵ-greedy approach. An ϵ-greedy approach offers several benefits over alternative exploration strategies:

(1) The amount of exploration can be manually tuned with the ϵ parameter, allowing either very conservative or very aggressive exploration.
(2) It is a priori known, via the ϵ parameter, how much performance we are giving up to perform exploration.
(3) The ϵ-greedy strategy generates stable and bounded propensity scores; this counteracts the high variance problem that is common in counterfactual learning and evaluation.

We consider two types of reward signals. First, the perfect scenario where a reward of 1 is given if the chosen label a_t is correct and 0 otherwise. Second, a near-random scenario where rewards are sampled from a Bernoulli distribution. More specifically, when a policy chooses the correct label a_t, the policy is given a reward of 1 with probability 0.6. Conversely, if the chosen label a_t is incorrect, the learner is given a reward of 1 with probability 0.4. We choose these contrasting scenarios as they highlight the need for safety. It is much easier for a learner to make mistakes in the near-random scenario due to the high levels of noise and we hypothesize that safety plays a more important role in that case. The text classification datasets that we use are specified in Table 1.

4.2 Document ranking task

For the *document ranking* task we use the counterfactual LTR framework as described in [23]. In this setup, a production ranker displays a ranked list to the user. It is assumed that the user does not

Table 2. Datasets used for the document ranking task.

Dataset	Queries	Features	Avg. docs per query
MSLR-WEB10K [37]	10,000	136	124
Yahoo Webscope [5]	36,251	700	23
Istella-s [33]	33,118	220	103

Table 3. Click noise settings for the document ranking task. Each entry is the probability of clicking a document given its relevance label.

	$P(click \mid relevance)$				
	0	1	2	3	4
Perfect	0.00	0.20	0.40	0.80	1.00
Position-biased	0.10	0.10	0.10	1.00	1.00
Near random	0.40	0.45	0.50	0.55	0.60

examine all documents in the presented ranking, but is instead more likely to observe top-ranked documents than lower ranked ones, a phenomenon called "position bias" [21]. After a user observes a document they can either judge it as relevant, by clicking on it, or judge it as non-relevant, by not clicking on it.

We simulate user behavior as follows:

(1) The policy being learned presents a ranked list to the simulated user.
(2) The simulated user samples a set of observed documents from the ranked list, where the probability of observing the document at rank i is

$$p_i = \left(\frac{1}{i}\right)^{\eta}, \tag{18}$$

where η is a parameter that controls the severity of click bias. This setup is identical to the one described in [23].
(3) For each observed document, we generate clicks depending on the relevance of the document. We use varying levels of click noise ("perfect", "position-biased" and "near-random"), the specific click probabilities are listed in Table 3. This setup is in line with the methodologies described in [13, 15, 34, 35].

In our setup, clicks are simulated only on the top 10 documents. This more realistically simulates the behavior of Web search users who are unlikely to visit the second (or later) result page [7]. For the counterfactual models we assume the correct propensity scores are known during learning. In other words, the propensity model used to simulate position bias is also used to compute the propensity scores during learning.

We update our model via stochastic gradient descent at the document level by updating the weights using a weighted gradient and stochastic gradient descent, as described in Section 2.2. Table 2 details the datasets used for the document ranking task.

4.3 Methods used for comparison

Our experiments are aimed at assessing the performance of SEA for solving contextual bandit problems. We compare SEA against offline and online methods. The former comparison is motivated by the fact that offline methods are safe by definition so a comparison of SEA against such methods will inform us about the potential performance gains by using SEA. The latter is motivated by the

fact that SEA is an online method so a comparison of SEA against such methods will inform us about the safety of using SEA. In all experiments, the models to be trained are warm-started with the weights of the deployed policy.

Offline methods for contextual bandit problems optimize a policy by observing actions that are taken by a separate logging policy π_b. This makes them safe, because learning happens only on data that has been collected in the past by a logging policy. However, safety also makes them incapable of exploration. We use the state-of-the-art offline method λ-translated Inverse Propensity Scoring (λ-IPS) [22]. We hypothesize that the average reward of a model trained by SEA is higher than of a model trained by λ-IPS because SEA is capable of exploring actions from the newly learned policy.

Online methods optimize a policy π_w by having it interact with the environment. Such methods are effective at exploring the action space but have no notion of safety. The online methods we use for the classification task are:

(1) policy gradient with an ϵ-greedy strategy,
(2) policy gradient with a Boltzmann exploration strategy,
(3) LinUCB [31], and
(4) Thompson Sampling [2].

For the ranking task, we use the following online methods:

(1) SVMRank Online [20]
(2) Dueling Bandit Gradient Descent [50] with Team-Draft Interleaving [38]

We hypothesize that the user experience of online methods suffers in the early stages of learning (i.e., performance will be below the baseline policy π_b) because these approaches do not provide formal guarantees of safety while exploring new actions.

Safe online methods are meant to be safe in the sense that during training, the algorithms never perform worse than a baseline policy π_b. Our contribution, SEA, falls in this class of methods. We also consider CLUCB (Conservative Linear UCB) [26]. Its design makes it impractical for problems with high dimensionality or large action spaces as it requires performing an $m \times m$ matrix inversion and solving a constrained optimization problem whenever an action has to be selected; its high computational complexity prevents us from using it with our datasets.

Finally, we also include an empirically safe version of SEA, which does not use the high-confidence bounds, which we denote as Boundless Safe Exploration Algorithm (BSEA). This method compares the mean estimate of the reward of the learned policy π_w and the deployed policy π_d, instead of the lower and upper confidence bounds. Specifically, we use Algorithm 1 where we set

$$UCB(\pi, \mathcal{D}_t) = LCB(\pi, \mathcal{D}_t) = \frac{1}{t} \sum_{i=1}^{t} \frac{r_{i,a_i}}{p_i} \pi(a_i \mid \boldsymbol{x}_i). \tag{19}$$

We expect this method to deploy its learned model earlier and more frequently than SEA because it does not have to overcome potentially large confidence bounds. Hence, we expect this policy to do better than SEA, but it may exhibit unsafe behavior as it no longer provides the same formal safety guarantees as SEA.

4.4 Choice of baseline policy

Both λ-IPS and SEA depend on a baseline policy π_b. We require that this baseline policy is sub-optimal, so that learning can occur. This requirement is reasonable since we cannot hope to improve an already optimal policy. We introduce suboptimality in π_b by subsampling the training set on which we train the policy (1% sample for the classification task and 0.1% sample for the ranking task). This is motivated by a scenario that commonly occurs in real search engines or classification systems: Manual labels are expensive to obtain and are usually available on a scale of several orders

of magnitude smaller than logged bandit feedback. This strategy for introducing suboptimality results in a baseline policy whose performance is much better than random, but still not optimal.

4.5 Metrics and statistical significance

We evaluate SEA and the competing approaches in two ways. One is in terms of *cumulative reward* during training, which is the sum of all rewards received as a function of the number of rounds; this type of metric allows us to quantify the degree to which the user experience is affected. A higher cumulative reward during training indicates a better user experience. The second is in terms of *average reward*, which is the reward averaged per action on held-out test data; this type of metric allows us to quantify how well a trained policy generalizes to unseen test data. The document ranking task uses simulated clicks as a reward signal (see Section 4.2) during training, which is not a very insightful metric for evaluating the true performance of a policy. Instead, to *evaluate* the learned policies, we use nDCG@10 [19]. We measure statistical significance of observed differences using a t-test ($p < 0.01$).

5 RESULTS

5.1 Safety

We first address **RQ1**:

Is SEA safe? I.e., does it always perform at least as good as a baseline policy?

To answer RQ1, we plot the performance of SEA against that of *online* algorithms while they are training, using average reward on a held-out test dataset as our metric.

Let us first consider the text classification task; see Figure 2. The baseline policies have an average accuracy of around 0.6, except in the 20 Newsgroups dataset where this is 0.4. The tested online algorithms include ϵ-greedy, Boltzmann exploration, Thompson sampling and LinUCB. For the 20 Newsgroups and RCV1 datasets we do not run LinUCB and Thompson sampling because they require inverting a 47,236 $\times$ 47,236 matrix (for 20 Newsgroups) and 62,060 $\times$ 62,060 matrix (for RCV1) on every update which is too computationally expensive. The performance of SEA is at least as good as the baseline policy. Similarly, it seems that warm-started online algorithms are also safe in this scenario, as they too perform always at least as good as the baseline policy. The best algorithm in terms of final performance is different for each of the datasets. Overall we find that when it is computationally feasible to apply UCB or Thompson sampling, they outperform all other methods. Furthermore we find that Boltzmann exploration works very well with a perfect reward signal. However, in the case of near random rewards, both Boltzmann exploration and ϵ-greedy are not always capable of learning a good policy. Finally, we observe that BSEA is on par with SEA, and in some cases significantly outperforms it, while being empirically safe. BSEA is capable of deploying its learned model faster and more frequently than SEA as it does not have to overcome potentially large confidence bounds. Although this means that BSEA is not guaranteed to be safe, we find that it is empirically safe across all experimental conditions.

Next, we consider the document ranking setting; see Figure 3. We use NDCG@10 on held-out test data as the evaluation metric. We observe that SEA is always at least as good as the baseline policy whereas online learning methods may suffer from unsafe performance in the early stages of learning. We see that in the case of near-random clicks on the Istella-s dataset, SEA accurately identifies that it is not safe to deploy whereas the online method suffers from unsafe performance and actually go below the baseline level. However, on the other datasets with near-random clicks, SEA never deploys a new model and consequently does not explore different actions. As a result, it is unable to improve upon the production policy. *Online* approaches outperform SEA here because they explore more aggressively and are able to learn even in the face of large amounts of noise.

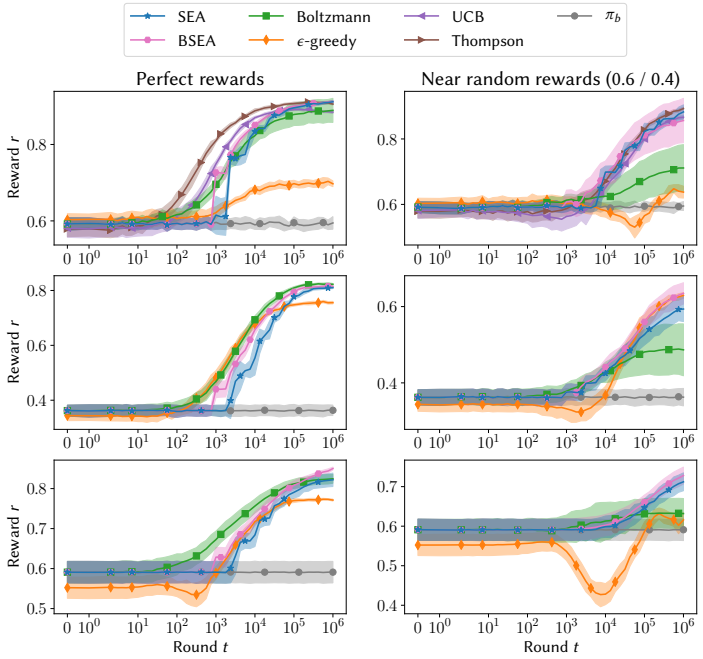

Fig. 2. The performance of SEA (blue line) compared to *online* algorithms for the text classification task. The top row indicates the USPS dataset, the middle row the 20 Newsgroups dataset and the bottom row the RCV1 dataset. The performance is measured on a held-out test set after each round t. Shaded areas indicate standard deviation. Best viewed in color.

These results are in line with previous work [18], which has shown that counterfactual methods have difficulty learning in scenarios with large amounts of noise. In cases with position-biased or perfect clicks, the ranker trained by SEA performs on par with the online method. This means that in realistic scenarios we do not sacrifice any performance by using SEA. Finally, we note that BSEA performs empirically safe on the document ranking task across all experimental settings, which is in line with the results on the text classification task.

This answers RQ1. Our experiments indicate that SEA is indeed safe, its performance does not fall below that of the baseline policy.

5.2 Improved user experience

Next, we answer **RQ2**:

> *Does SEA provide a better user experience during training than offline or online learning methods? I.e., does it accumulate a higher reward during training?*

To answer RQ2, we measure the cumulative reward obtained by the learning algorithms. During training, the offline method, λ-IPS, only observes actions taken by the baseline policy, hence its cumulative reward is always equal to the cumulative reward of the baseline and is omitted from the result tables to save space.

Table 4 lists the results for the text classification task. The cumulative reward achieved by SEA is at least as good as the baseline, and eventually better. We find that only ϵ-greedy on the RCV1 dataset under near random rewards performs significantly worse than the baseline. In all other settings we find that all the warm-started online methods similarly provide a user experience that is

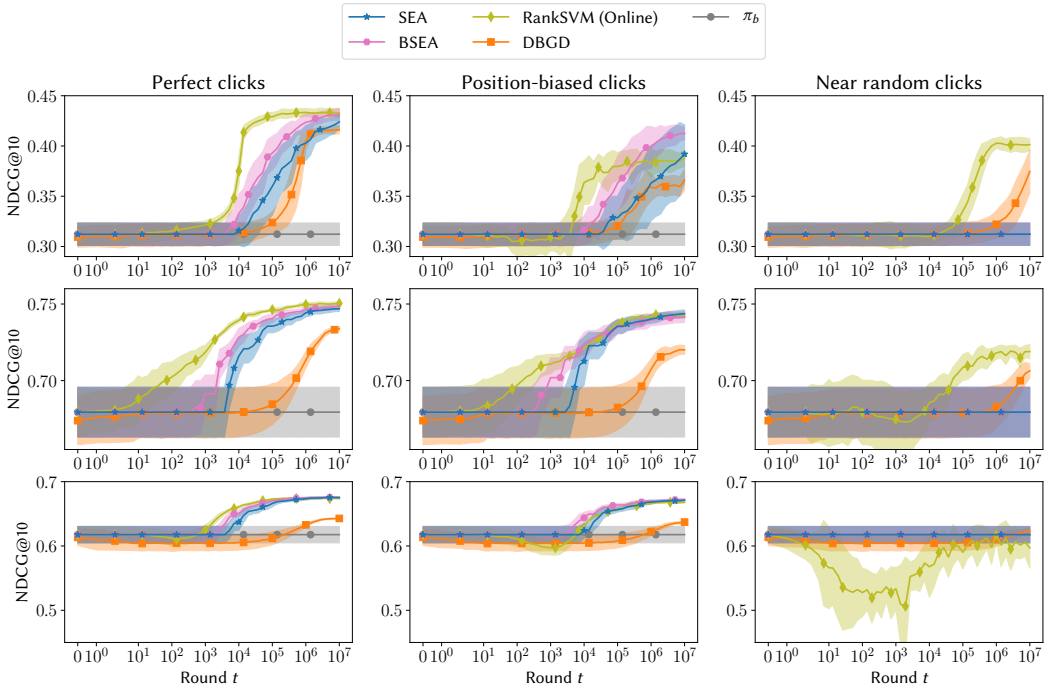

Fig. 3. The performance of SEA (blue line) compared to an *online* approach (orange line) for the document ranking task on MSLR-10k (top row), Yahoo Webscope (middle row) and Istella-s (bottom row) with varying levels of click noise. The performance of the trained ranker is measured on a held-out test set after each round t. Shaded areas indicate standard deviation. Best viewed in color.

at least as good as the baseline, and eventually better. This means that for the text classification task, all methods, with the exception of ϵ-greedy, perform safely and never harm the user experience. Finally, we see that BSEA is able to improve the user experience faster and more quickly than SEA in the perfect rewards setting, and performs on par in the near random rewards setting. Similar to the results we have observed in Section 5.1, we hypothesize that BSEA does not have to overcome potentially large confidence bounds and as a result is able to more quickly and more frequently deploy its learned model. As a result, BSEA is capable of improving the user experience over SEA.

Next we turn to the case of ranking. See Table 5 for the cumulative reward results for document ranking. It is clear that SEA performs at least as good as the baseline policy and eventually outperforms it. For the online learning method, this is not the case. Specifically for the Istella-s dataset with near random clicks, we observe significant performance degradations in the early stages of learning which harms the user experience. However, on the other datasets with near random clicks we find that the online methods are capable of safely exploring the action space, even with high noise, and as a result improve the overall user experience more than SEA. From Table 6 we see that Istella-s contains very sparse relevance feedback compared to the MSLR10k and Webscope datasets. In the near-random click scenario this means that there are significantly more clicks on non-relevant documents, making Istella-s with near-random clicks the most challenging learning scenario. We once again see that BSEA is a very strong baseline across all datasets, improving the user experience over SEA in the perfect and position-biased clicks settings while performing on par with SEA in the near-random click setting.

Table 4. Cumulative reward, relative to the baseline policy, while training for text classification. LinUCB and Thompson Sampling cannot be run on the 20 Newsgroups and RCV1 datasets due to their high computational complexity. Statistically significant differences with SEA are indicated using $^{\blacktriangle}$ ($p < 0.01$) for gains and $^{\blacktriangledown}$ ($p < 0.01$) for losses.

	Round t	10^2	10^3	10^4	10^5	10^6
				Perfect rewards		
USPS	UCB	1.73	105.53$^{\blacktriangle}$	2568.33$^{\blacktriangle}$	32496.67$^{\blacktriangle}$	350226.27$^{\blacktriangle}$
	Thompson	4.47	172.40$^{\blacktriangle}$	2833.00$^{\blacktriangle}$	33298.33$^{\blacktriangle}$	352940.07$^{\blacktriangle}$
	ϵ-greedy	0.80	31.87	699.07$^{\blacktriangledown}$	10568.53$^{\blacktriangledown}$	116854.20$^{\blacktriangledown}$
	Boltzmann	0.93	69.27$^{\blacktriangle}$	1950.47	28002.60	313452.47
	BSEA	0.00	0.00	2086.93$^{\blacktriangle}$	30483.40$^{\blacktriangle}$	341919.60$^{\blacktriangle}$
	SEA	0.00	0.00	1720.73	29286.93	337577.53
20-News	ϵ-greedy	-2.00	60.07$^{\blacktriangle}$	2571.53$^{\blacktriangle}$	42913.73$^{\blacktriangle}$	499605.53$^{\blacktriangledown}$
	Boltzmann	0.33	56.47$^{\blacktriangle}$	2388.53$^{\blacktriangle}$	45220.73$^{\blacktriangle}$	564285.20$^{\blacktriangle}$
	BSEA	0.00	0.00	1935.87$^{\blacktriangle}$	43151.33$^{\blacktriangle}$	564200.73$^{\blacktriangle}$
	SEA	0.00	0.00	1083.47	37342.53	540070.47
RCV1	ϵ-greedy	-3.53	-28.33	905.40$^{\blacktriangle}$	21396.73$^{\blacktriangle}$	283490.73
	Boltzmann	0.80	51.40	1356.07$^{\blacktriangle}$	22085.53$^{\blacktriangle}$	269428.07
	BSEA	0.00	0.00	946.67	22094.07$^{\blacktriangle}$	322481.00$^{\blacktriangle}$
	SEA	0.00	0.00	573.40	18856.20	282126.73
				Near random rewards (0.6 / 0.4)		
USPS	UCB	-1.73	3.93	89.00	3617.53	59778.47
	Thompson	-1.73	-3.47	121.67	4024.93	61446.93
	ϵ-greedy	-0.60	9.73	52.87	-534.00$^{\blacktriangledown}$	7380.40$^{\blacktriangledown}$
	Boltzmann	0.33	3.20	52.00	1297.73$^{\blacktriangledown}$	23223.27$^{\blacktriangledown}$
	BSEA	0.00	0.00	87.53	3687.80	51662.27
	SEA	0.00	0.00	50.40	3567.00	55458.73
20-News	ϵ-greedy	-1.87	-11.33	-20.67	2440.47	56843.47
	Boltzmann	-2.40	-2.67	95.40	2183.27	32669.13
	BSEA	-0.33	1.60	108.67	3015.47	58141.20
	SEA	-0.33	1.60	82.53	2534.20	48437.53
RCV1	ϵ-greedy	-0.47	-9.40	-253.80$^{\blacktriangledown}$	-1362.07$^{\blacktriangledown}$	5222.00$^{\blacktriangledown}$
	Boltzmann	-0.33	0.80	27.53	807.40	8887.93$^{\blacktriangledown}$
	BSEA	0.00	0.47	0.93	905.87	25312.67
	SEA	0.00	0.00	7.60	706.60	21776.53

This answers RQ2. SEA provides a better user experience than online methods, particularly in the document ranking setting with high levels of click noise where the user experience may be negatively affected by online algorithms. In the later stages of learning SEA provides a user experience that is significantly better than the baseline and comparable to online approaches. We find that BSEA is able to improve the user experience even further by being able to deploy its learned model earlier and more frequently.

Table 5. Cumulative NDCG@10, relative to the baseline policy, while training for document ranking under varying levels of click noise. Statistical significance is denoted in the same way as in Table 4.

	Round t	10^2	10^3	10^4	10^5	10^6	10^7
				Perfect clicks			
MSLR10k	RankSVM (Online)	0.43	6.48	259.78$^{\blacktriangle}$	10065.58$^{\blacktriangle}$	115823.32$^{\blacktriangle}$	1180033.12$^{\blacktriangle}$
	DBGD	-0.05	-0.62	-1.22	541.43	48244.85$^{\blacktriangle}$	940724.90$^{\blacktriangle}$
	BSEA	0.00	0.00	49.80	5513.30$^{\blacktriangle}$	95560.77$^{\blacktriangle}$	1125341.57$^{\blacktriangle}$
	SEA	0.00	0.00	6.52	2724.04	71507.83$^{\blacktriangle}$	996331.68$^{\blacktriangle}$
Webscope	RankSVM (Online)	1.81	31.75$^{\blacktriangle}$	514.23$^{\blacktriangle}$	6477.67$^{\blacktriangle}$	70834.57$^{\blacktriangle}$	730703.57$^{\blacktriangle}$
	DBGD	-0.14	-1.22	-4.49	222.10	19343.31	475561.27$^{\blacktriangle}$
	BSEA	-0.01	2.98	325.39$^{\blacktriangle}$	5518.60$^{\blacktriangle}$	64913.61$^{\blacktriangle}$	687719.69$^{\blacktriangle}$
	SEA	-0.01	-0.05	151.24	4661.96$^{\blacktriangle}$	60199.38$^{\blacktriangle}$	670419.85$^{\blacktriangle}$
Istella-s	RankSVM (Online)	0.05	0.96	325.50$^{\blacktriangle}$	5364.95$^{\blacktriangle}$	61424.60$^{\blacktriangle}$	631101.11$^{\blacktriangle}$
	DBGD	-1.27	-13.16	-130.50	-911.40	6156.68	222276.99$^{\blacktriangle}$
	BSEA	0.00	0.00	206.87	4749.66$^{\blacktriangle}$	59609.20$^{\blacktriangle}$	634840.28$^{\blacktriangle}$
	SEA	0.00	0.00	76.25	3982.61$^{\blacktriangle}$	56417.23$^{\blacktriangle}$	624784.49$^{\blacktriangle}$
				Position-biased clicks			
MSLR10k	RankSVM (Online)	-0.23	-5.17	184.43	5845.83$^{\blacktriangle}$	68623.01$^{\blacktriangle}$	725200.78$^{\blacktriangle}$
	DBGD	-0.06	-0.81	-3.12	368.26	30558.36$^{\blacktriangle}$	453895.21$^{\blacktriangle}$
	BSEA	0.00	0.00	3.85	2849.70	69915.76$^{\blacktriangle}$	927569.46$^{\blacktriangle}$
	SEA	0.00	0.00	-0.01	856.81	32375.36	634675.69$^{\blacktriangle}$
Webscope	RankSVM (Online)	1.04	26.88	349.04$^{\blacktriangle}$	4960.61$^{\blacktriangle}$	61055.11$^{\blacktriangle}$	642520.10$^{\blacktriangle}$
	DBGD	-0.15	-1.15	-6.12	96.66	14787.60	353452.62$^{\blacktriangle}$
	BSEA	0.00	8.86	324.38$^{\blacktriangle}$	4920.24$^{\blacktriangle}$	57553.64$^{\blacktriangle}$	614338.66$^{\blacktriangle}$
	SEA	0.00	0.01	144.06	4497.80$^{\blacktriangle}$	57702.23$^{\blacktriangle}$	633904.88$^{\blacktriangle}$
Istella-s	RankSVM (Online)	-0.08	-13.49	-14.85	3303.76$^{\blacktriangle}$	50451.05$^{\blacktriangle}$	561270.85$^{\blacktriangle}$
	DBGD	-1.27	-13.14	-133.17	-1132.55	-1749.76	150641.69
	BSEA	0.00	0.00	128.66	4145.78$^{\blacktriangle}$	54614.90$^{\blacktriangle}$	603275.09$^{\blacktriangle}$
	SEA	0.00	0.00	0.00	3165.49$^{\blacktriangle}$	50018.04$^{\blacktriangle}$	581799.13$^{\blacktriangle}$
				Near random clicks			
MSLR10k	RankSVM (Online)	0.11	-0.71	-24.48	850.82	66464.76$^{\blacktriangle}$	849713.67$^{\blacktriangle}$
	DBGD	-0.06	-0.78	-7.48	-6.20	4004.35	339609.56$^{\blacktriangle}$
	BSEA	0.00	-0.00	-0.01	-0.06	-0.09	0.84
	SEA	0.00	-0.00	-0.01	-0.06	-0.09	0.84
Webscope	RankSVM (Online)	0.23	-2.44	1.81	1829.45	34646.76$^{\blacktriangle}$	403139.88$^{\blacktriangle}$
	DBGD	-0.14	-1.24	-7.46	-50.84	1100.02	169029.43
	BSEA	0.01	0.03	0.03	-0.12	0.83	1.18
	SEA	0.01	0.03	0.03	-0.12	0.83	1.18
Istella-s	RankSVM (Online)	-7.09	-83.71$^{\blacktriangledown}$	-678.03$^{\blacktriangledown}$	-3210.06$^{\blacktriangledown}$	-17794.95	-138288.71
	DBGD	-1.27	-13.07	-135.09	-1284.11	-10421.04	3837.78
	BSEA	0.00	0.00	0.00	0.00	0.00	0.00
	SEA	0.00	0.00	0.00	0.00	0.00	0.00

Table 6. Distribution of relevance grades on documents for the ranking datasets. Note that Istella-s is very sparse, nearly 90% of the documents are judged as non-relevant.

Relevance grade	0	1	2	3	4
Webscope	0.26	0.36	0.28	0.08	0.02
MSLR10k	0.52	0.32	0.13	0.02	0.01
Istella-s	0.89	0.02	0.04	0.03	0.02

Table 7. Average reward on held-out test data for the learned model for the document classification task after 1,000,000 rounds. Statistical significance is denoted the same way as in Table 4.

		Perfect rewards	*Near random rewards (0.6 / 0.4)*
USPS	ϵ-greedy	0.90$^{\blacktriangledown}$	0.82$^{\blacktriangledown}$
	Boltzmann	0.90	0.72$^{\blacktriangledown}$
	UCB	0.91$^{\blacktriangledown}$	0.91
	Thompson	0.91$^{\blacktriangledown}$	0.91
	IPS	0.92	0.89
	BSEA	0.92	0.86
	SEA	0.92	0.90
20-News	ϵ-greedy	0.83	0.69
	Boltzmann	0.88$^{\blacktriangle}$	0.53$^{\blacktriangledown}$
	IPS	0.83$^{\blacktriangledown}$	0.53$^{\blacktriangledown}$
	BSEA	0.86	0.74
	SEA	0.85	0.70
RCV1	ϵ-greedy	0.85	0.68
	Boltzmann	0.83	0.63$^{\blacktriangledown}$
	IPS	0.81	0.68
	BSEA	0.87$^{\blacktriangle}$	0.74
	SEA	0.84	0.73

5.3 The benefit of exploring

Finally, we turn to **RQ3**:

> *Does SEA, which explores the action space, learn a more effective policy compared to offline learning methods, which do not perform exploration?*

To answer RQ3, we compare SEA to the state-of-the-art offline learning algorithm λ-IPS on unseen test data. The results for the classification task are displayed in Table 7. Because SEA is capable of exploring, it finds highly favorable regions of the action space. λ-IPS is, by design, incapable of exploration and cannot deviate far from the baseline policy in terms of performance. As a result, in the case of text classification, the final model learned by SEA outperforms the final model learned by λ-IPS on the 20 Newsgroups datasets. On the USPS and RCV1 dataset, there is no noticeable performance difference between λ-IPS and SEA. Note that the baseline policy for USPS and RCV1 has a performance of around 0.6, whereas the baseline policy for the 20 Newsgroups dataset only has a performance of around 0.4. We postulate that this difference in baseline performance may cause λ-IPS to learn better and therefore perform equally to SEA on the USPS and RCV1 datasets. Lastly, we find that BSEA performs on par with SEA in all settings, while producing a significantly better model on the RCV1 dataset with perfect rewards. Similar to our observations in Sections 5.1

Table 8. Average reward on held-out test data for the learned model for the document ranking task after 10,000,000 rounds. Statistical significance is denoted the same way as in Table 4.

		Perfect clicks	*Position-biased clicks*	*Near random clicks*
MSLR10k	RankSVM (Online)	0.43	0.39	0.40$^{\blacktriangle}$
	DBGD	0.43	0.38	0.39$^{\blacktriangle}$
	IPS	0.35$^{\blacktriangledown}$	0.35$^{\blacktriangledown}$	0.32
	BSEA	0.43	0.41	0.32
	SEA	0.43	0.40	0.32
Webscope	RankSVM (Online)	0.75	0.74	0.72$^{\blacktriangle}$
	DBGD	0.74$^{\blacktriangledown}$	0.72$^{\blacktriangledown}$	0.71
	IPS	0.73$^{\blacktriangledown}$	0.74$^{\blacktriangledown}$	0.69
	BSEA	0.75	0.74	0.69
	SEA	0.75	0.74	0.69
Istella-s	RankSVM (Online)	0.67	0.67$^{\blacktriangledown}$	0.60
	DBGD	0.66$^{\blacktriangledown}$	0.65$^{\blacktriangledown}$	0.64
	IPS	0.67$^{\blacktriangledown}$	0.66$^{\blacktriangledown}$	0.64
	BSEA	0.68	0.67	0.64
	SEA	0.68	0.67	0.64

and 5.2, we find that BSEA likely performs so well because it does not have to overcome confidence bounds and can deploy its learned model faster and more frequently than SEA, allowing it to learn a more effective model.

Finally, we consider the document ranking setting; see Table 8. SEA is able to learn a more effective ranker than λ-IPS on all datasets with perfect clicks and on Istella-s with position-biased clicks. We hypothesize that this is because SEA, being capable of exploration, eventually shows documents to the user that the baseline policy would rarely, if ever, show. As a result, SEA is able to obtain clicks on these documents, allowing it to learn more effectively. This is in line with our expectations because previous work has shown that even a tiny amount of exploration can result in substantial improvements in LTR [14]. Finally, we find that BSEA does not produce a better ranker than SEA for the document ranking task. Across all settings, the models produced by BSEA and SEA are comparable.

This answers RQ3. Exploration does indeed help the performance of the policy learned by SEA and it outperforms λ-IPS both for scenarios with a small action space (e.g., text classification) and for scenarios with a large action space (e.g., document ranking).

6 RELATED WORK

The idea of deploying automated decision making systems in IR is not new. The contextual bandit framework has been used in news recommendation [31], ad placement [28], and online learning to rank [13]. Contextual bandit formulations of IR problems allow insights and methods from the bandit literature to be applied and extended to address these problems [14]. For research on contextual bandits this connection has opened up an application area where new approaches can be evaluated on large-scale datasets [32].

6.1 Learning in contextual bandits

Contextual bandit algorithms have been widely studied in the *online* and *offline* learning settings [2, 31, 43]. A key challenge in contextual bandit problems is the exploration-vs-exploitation tradeoff. On the one hand we want to explore new actions so as to find favorable rewards. On the other hand, we want to exploit existing knowledge about actions so as to maximize the total reward. The online learning methods we consider in this paper deal with this tradeoff in various ways. The methods we consider are policy gradient with ϵ-greedy exploration, policy gradient with Boltzmann exploration, LinUCB and Thompson Sampling. Policy gradient methods optimize the weights of a policy via stochastic gradient descent, by solving the following optimization problem:

$$\min_{\pi_w} -\sum_{t=1}^{T} \log\left(\pi_w(a_t \mid \boldsymbol{x}_t)\right) \cdot r_t. \tag{20}$$

The ϵ-greedy heuristic selects actions by choosing with probability ϵ an action uniformly at random and with probability $(1-\epsilon)$ the best possible action. Boltzmann exploration [4] chooses actions by drawing them with a probability proportional to the policy: $a_t \sim \pi_w(\cdot \mid \boldsymbol{x}_t)$. LinUCB [31] and Thompson Sampling [2] are different from policy gradient methods because they make a linearity assumption. These methods construct a set of weights $w_a \in \mathbb{R}^m$ for every possible action, such that $\pi_w(a_t \mid \boldsymbol{x}_t) = w_{a_t}^T \boldsymbol{x}_t$. LinUCB selects actions by choosing the one with the highest confidence bound. Thompson Sampling samples new weights $\hat{w}_a$ from a posterior distribution and then chooses the best action given the sampled weights: $a_t = \arg\max_a \pi_{\hat{w}_a}(a \mid \boldsymbol{x}_t)$.

In IR there has been considerable attention for exploiting log data [12]. It is one of the most ubiquitous forms of data available, as it can be recorded from a variety of systems at little cost [42]. The interaction logs of such systems typically contain a record of the input to the system, the prediction made by the system, and the feedback. The feedback provides only partial information limited to the particular prediction shown by the system. Offline learning algorithms tell us how data collected from interaction logs of one system can be used to optimize a new system [14, 42]. Interaction logs used in offline learning are usually biased towards the policy that collected the data. To remove this bias, offline learning methods resort to inverse propensity scoring. To use inverse propensity scoring, the logging policy must be stochastic and the corresponding propensity scores (probability of choosing the logged action) are recorded. Using these propensity scores, it is possible to reweigh data samples to remove the bias. An advantage of offline learning methods is that they do not require interactive experimental control. Thus, there is no risk of hurting the user experience with these methods. In other words, offline learning methods are safe by definition [18].

Unlike *online* methods, SEA's performance during the early stages of learning does not suffer and always stays at least as good as a baseline, making the method safe to use. Compared to *offline* methods, SEA is capable of exploration, which makes it effective at finding areas of the action space that may have high reward.

6.2 Safety in contextual bandits

There has been a growing interest in concepts related to safety for contextual bandit problems. This is due to the fact that contextual bandit formulations are applied to automated decision making systems, where actions taken by the system can have a significant impact on the real world. There are two main groups of work: *risk-aware* methods and *conservative* methods.

Risk-aware methods [8, 17, 41] are online learning algorithms that model the risk associated with executing certain actions as a cost which is to be constrained and minimized. Galichet et al. [8] introduce the concept of risk-awareness for the multi-armed bandit framework. Sun et al. [41] extend the idea of risk-awareness to the adversial contextual bandit setting. Garcia and Fernández

[9] explore risk-awareness for the reinforcement learning paradigm. Risk-aware methods use a separate type of feedback signal, in addition to the standard reward feedback, which is called *risk*. Risk-aware methods aim to keep the cumulative risk below a specific threshold. These types of methods are typically applied in fields like robotics where certain actions can be dangerous or cause damage. A drawback is that the risk has to be explicitly quantified by the environment. Designing a good risk feedback signal for IR systems is subject to modeling biases and in some cases impossible, limiting the application of such methods.

Conservative methods [49] measure safety as a policy's performance relative to a baseline policy. A method is safe if its performance is always within some margin of the baseline policy. The idea was first introduced for the multi-armed bandit case [49] and later extended to linear contextual bandits in the form of the CLUCB algorithm [26]. CLUCB is a safe conservative online learning method, which works by constructing confidence sets around the parameters of the policy. Unfortunately, the method has to solve a constrained optimization problem every time an action has to be selected, which has a significant computational overhead. In contrast, SEA addresses this problem because it only needs to do two computationally efficient operations: update a lower confidence bound estimate and perform a gradient update step. This makes SEA applicable to larger and more complex datasets.

Finally, Li et al. [30] introduce a complementary approach to SEA, called BubbleRank, an algorithm that gradually improves upon an initial ranked list by exchanging higher-ranked less attractive items for lower- ranked more attractive items. Li et al. define a safety constraint that is based on incorrectly-ordered item pairs in the ranked list, and prove that BubbleRank never violates this constraint with a high probability. We do not compare to BubbleRank in our experiments because it assumes a user model [6], an assumption that is orthogonal to our experimental setup [18].

7 CONCLUSION

We have proposed SEA, a *safe online* learning algorithm for contextual bandit problems. SEA learns a new policy from the behavior of an existing baseline policy and then starts to execute actions from the new policy once its estimated performance is at least as good as that of the baseline. This brings us the best of both worlds, achieving the performance of *online* learning with the safety of *offline* learning.

We perform extensive experimentation on two IR tasks, *text classification* and *document ranking*. In both tasks SEA is safe. It never performs worse than a baseline policy, whereas online methods are unsafe and suffer from suboptimal performance in the early stages of learning. We observe that the user experience with SEA is improved in the early stages of training, but may be suboptimal in later stages when compared to methods that converge much faster, such as LinUCB (although such methods are not generalizable to all datasets). The final performance of a model learned with SEA is comparable to other online algorithms and beats that of offline methods, which are incapable of exploration. Finally, we find that BSEA, a boundless version of SEA, is empirically just as safe as SEA while being able to explore faster and, as a result, outperform SEA in many experimental conditions. These results confirm that SEA does indeed trade off advantages and disadvantages of *offline* and *online* learning, in some scenarios outperforming online methods in the early stages of learning and having higher final performance than offline methods. However, compared to purely *online* approaches, SEA may not be as effective in learning a good policy due to the fact that it is more conservative when exploring. The conservative nature of SEA implies that safety is not free, there is a possible performance cost involved for cases where SEA is unable to effectively explore.

An interesting direction for future work is an extension of SEA to non-linear models. SEA builds on gradient descent and it is trivial to extend the method to use gradient-boosted decision trees or

neural networks. This line of work is especially applicable for the document ranking task where it is known that non-linear models can outperform linear models by a wide margin [48]. Furthermore, another possibility for future work is to perform a study on safety for a broad range of online and offline methods, similar to [18]. Specifically, it would be interesting to compare against recent work on safe online learning to re-rank [30].

CODE AND DATA

To facilitate reproducibility of the results in this paper, we are sharing all resources used in this paper at https://github.com/rjagerman/tois2020-safe-exploration-algorithm. This includes the training procedures and parameters of the online, offline, safe and baseline policies.

ACKNOWLEDGMENTS

We thank Chang Li and Harrie Oosterhuis for valuable discussions. We thank our anonymous reviewers for inspiring questions and helpful suggestions.

This research was partially supported by Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), the Innovation Center for Artificial Intelligence (ICAI), and the Netherlands Organization for Scientific Research (NWO) under project number 612.001.551.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)* 23, 1 (2005), 103–145.

[2] Shipra Agrawal and Navin Goyal. 2013. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*. 127–135.

[3] Alina Beygelzimer and John Langford. 2009. The offset tree for learning with partial labels. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 129–138.

[4] Nicolò Cesa-Bianchi, Claudio Gentile, Gergely Neu, and Gabor Lugosi. 2017. Boltzmann exploration done right. In *Advances in Neural Information Processing Systems*. 6275–6284.

[5] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*. 1–24.

[6] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. Morgan & Claypool Publishers.

[7] Aleksandr Chuklin, Pavel Serdyukov, and Maarten de Rijke. 2013. Modeling clicks beyond the first result page. In *Proceedings of the 22nd ACM Conference on Information and Knowledge Management*. ACM, 1217–1220.

[8] Nicolas Galichet, Michele Sebag, and Olivier Teytaud. 2013. Exploration vs exploitation vs safety: Risk-aware multi-armed bandits. In *Proceedings of the 5th Asian Conference on Machine Learning*. PMLR, 245–260.

[9] Javier Garcia and Fernando Fernández. 2012. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research* 45 (2012), 515–564.

[10] Dorota Glowacka. 2019. Bandit algorithms in information retrieval. *Foundations and Trends in Information Retrieval* 13, 4 (2019), 299–424.

[11] Artem Grotov, Shimon Whiteson, and Maarten de Rijke. 2015. Bayesian ranker comparison based on historical user interactions. In *Proceeedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 273–282.

[12] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. 2013. Reusing historical interaction data for faster online learning to rank for information retrieval. In *WSDM 2013: International Conference on Web Search and Data Mining*. ACM, 183–192.

[13] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2011. Balancing exploration and exploitation in learning to rank online. In *Advances in Information Retrieval – 33rd European Conference on IR Research*. Springer, 251–263.

[14] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2011. Contextual bandits for information retrieval. In *NIPS 2011 Workshop on Bayesian Optimization, Experimental Design, and Bandits*.

[15] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2013. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Transactions on Information Systems (TOIS)* 31, 4 (2013), 17.
[16] Jonathan J. Hull. 1994. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16, 5 (1994), 550–554.
[17] Xiaoguang Huo and Feng Fu. 2017. Risk-aware multi-armed bandit problem with application to portfolio selection. *arXiv preprint arXiv:1709.04415* (2017).
[18] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To model or to intervene: A comparison of counterfactual and online learning to rank from user interactions. In *Proceedings of the 42nd international ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 15–24.
[19] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.
[20] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 133–142.
[21] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*. ACM, 154–161.
[22] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. 2018. Deep learning from logged bandit feedback. In *International Conference on Learning Representations*.
[23] 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*. ACM, 781–789.
[24] Leslie Pack Kaelbling. 1994. Associative reinforcement learning: Functions in k-DNF. *Machine Learning* 15, 3 (1994), 279–298.
[25] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. 2008. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, 440–447.
[26] Abbas Kazerouni, Mohammad Ghavamzadeh, and Benjamin Van Roy. 2016. Conservative contextual linear bandits. *arXiv preprint arXiv:1611.06426* (2016).
[27] Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, Vol. 10. 331–339.
[28] John Langford, Alexander Strehl, and Jennifer Wortman. 2008. Exploration scavenging. In *Proceedings of the 25th International Conference on Machine learning*. ACM, 528–535.
[29] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research* 5, Apr (2004), 361–397.
[30] Chang Li, Branislav Kveton, Tor Lattimore, Ilya Markov, Maarten de Rijke, Csaba Szepesvari, and Masrour Zoghi. 2019. BubbleRank: Safe online learning to re-rank via implicit click feedback. In *UAI 2019: Conference on Uncertainty in Artificial Intelligence*.
[31] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 661–670.
[32] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM International Conference on Web search and Data Mining*. ACM, 297–306.
[33] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. 2016. Post-learning optimization of tree ensembles for efficient ranking. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 949–952.
[34] Harrie Oosterhuis and Maarten de Rijke. 2017. Balancing speed and quality in online learning to rank for information retrieval. In *Proceedings of the 26th ACM Conference on Information and Knowledge Management*. ACM, 277–286.
[35] Harrie Oosterhuis, Anne Schuth, and Maarten de Rijke. 2016. Probabilistic multileave gradient descent. In *Advances in Information Retrieval – 38th European Conference on IR Research*. Springer, 661–668.
[36] Art B Owen. 2013. Monte Carlo theory, methods and examples. *Monte Carlo Theory, Methods and Examples. Art Owen* (2013).
[37] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. *arXiv preprint arXiv:1306.2597* (2013).
[38] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. 2008. How Does Clickthrough Data Reflect Retrieval Quality?. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM '08)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/1458082.1458092
[39] Alex Strehl, John Langford, Lihong Li, and Sham M Kakade. 2010. Learning from logged implicit exploration data. In *Advances in Neural Information Processing Systems*. 2217–2225.
[40] Alexander L Strehl, Chris Mesterharm, Michael L Littman, and Haym Hirsh. 2006. Experience-efficient learning in associative bandit problems. In *Proceedings of the 23rd International Conference on Machine learning*. ACM, 889–896.

[41] Wen Sun, Debadeepta Dey, and Ashish Kapoor. 2016. Risk-aware algorithms for adversarial contextual bandits. *arXiv preprint arXiv:1610.05129* (2016).
[42] Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research* 16, 1731–1755 (2015).
[43] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual risk minimization: Learning from logged bandit feedback. In *International Conference on Machine Learning*. 814–823.
[44] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miroslav Dudík, John Langford, Damien Jose, and Imed Zitouni. 2016. Off-policy evaluation for slate recommendation. *arXiv preprint arXiv:1605.04812* (2016).
[45] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. 2015. High-confidence off-policy evaluation. In *AAAI*. 3000–3006.
[46] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 610–618.
[47] Zeng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Reinforcement learning to rank with Markov decision process. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 945–948.
[48] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* 13, 3 (2010), 254–270.
[49] Yifan Wu, Roshan Shariff, Tor Lattimore, and Csaba Szepesvári. 2016. Conservative bandits. In *International Conference on Machine Learning*. 1254–1262.
[50] 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 (ICML '09)*. ACM, New York, NY, USA, 1201–1208. https://doi.org/10.1145/1553374.1553527