# Cascading Non-Stationary Bandits: Online Learning to Rank in the Non-Stationary Cascade Model

**Chang Li** and **Maarten de Rijke**
University of Amsterdam
{c.li, derijke}@uva.nl

## Abstract

Non-stationarity appears in many online applications such as web search and advertising. In this paper, we study the online learning to rank problem in a *non-stationary* environment where user preferences change abruptly at an unknown moment in time. We consider the problem of identifying the $K$ most attractive items and propose *cascading non-stationary bandits*, an online learning variant of the *cascading model*, where a user browses a ranked list from top to bottom and clicks on the first attractive item. We propose two algorithms for solving this non-stationary problem: CascadeDUCB and CascadeSWUCB. We analyze their performance and derive gap-dependent upper bounds on the $n$-step regret of these algorithms. We also establish a lower bound on the regret for cascading non-stationary bandits and show that both algorithms match the lower bound up to a logarithmic factor. Finally, we evaluate their performance on a real-world web search click dataset.

## 1 Introduction

Learning to rank LTR [Liu, 2009] is a combination of machine learning and information retrieval. It is a core problem in many applications, such as web search and recommendation [Liu, 2009; Zoghi *et al.*, 2017]. The goal of LTR is to rank items, e.g., documents, and show the top $K$ items to a user. Traditional LTR algorithms are supervised, offline algorithms; they learn rankers from human annotated data [Qin *et al.*, 2010] and/or users' historical interactions [Joachims, 2002]. Every day billions of users interact with modern search engines and leave a trail of interactions. It is feasible and important to design online algorithms that directly learn from such user clicks to help improve users' online experience. Indeed, recent studies show that even well-trained production rankers can be optimized by using users' online interactions, such as clicks [Zoghi *et al.*, 2016].

Generally, interaction data is noisy [Joachims, 2002], which gives rise to the well-known exploration vs. exploitation dilemma. Multi-armed bandit (MAB) [Auer *et al.*, 2002] algorithms have been designed to balance exploration and exploitation. Based on MABs, many online LTR algorithms have been published [Radlinski *et al.*, 2008; Kveton *et al.*, 2015; Katariya *et al.*, 2016; Lagrée *et al.*, 2016; Zoghi *et al.*, 2017; Li *et al.*, 2019]. These algorithms address the exploration vs. exploitation dilemma in an elegant way and aim to maximize user satisfaction in a stationary environment where users do not change their preferences over time. Moreover, they often come with regret bounds.

Despite the success of the algorithms mentioned above in the stationary case, they may have linear regret in a non-stationary environment where users may change their preferences abruptly at any unknown moment in time. Non-stationarity widely exists in real-world application domains, such as search engines and recommender systems [Yu and Mannor, 2009; Pereira *et al.*, 2018; Wu *et al.*, 2018; Jagerman *et al.*, 2019]. Particularly, we consider *abruptly changing environments* where user preferences remain constant in certain time periods, named *epochs*, but change occurs abruptly at unknown moments called *breakpoints*. The abrupt changes in user preferences give rise to a new challenge of balancing "remembering" and "forgetting" [Besbes *et al.*, 2014]: the more past observations an algorithm retains the higher the risk of making a biased estimator, while the fewer observations retained the higher stochastic error it has on the estimates of the user preferences.

In this paper, we propose *cascading non-stationary bandits*, an online variant of the cascade model (CM) [Craswell *et al.*, 2008] with the goal of identifying the $K$ most attractive items in a non-stationary environment. CM is a widely-used model of user click behavior [Chuklin *et al.*, 2015; Zoghi *et al.*, 2017]. In CM, a user browses the ranked list from top to bottom and clicks the first attractive item. The items ranked above the first clicked item are browsed but not attractive since they are not clicked. The items ranked below the first clicked item are not browsed since the user stops browsing the ranked list after a click. Although CM is a simple model, it effectively explains user behavior [Kveton *et al.*, 2015].

Our key technical contributions in this paper are: (1) We formalize a non-stationary online learning to rank (OLTR) problem as cascading non-stationary bandits. (2) We propose two algorithms, CascadeDUCB and CascadeSWUCB, for solving it. They are motivated by *discounted UCB* (DUCB) and *sliding window UCB* (SWUCB), respectively [Garivier and Moulines, 2011]. CascadeDUCB balances "remembering" and "forgetting" by using a discounting factor of past observations, and

CascadeSWUCB balances the two by using statistics inside a fixed-size sliding window. (3) We derive gap-dependent upper bounds on the regret of the proposed algorithms. (4) We derive a lower bound on the regret of cascading non-stationary bandits. We show that the upper bounds match this lower bound up to a logarithmic factor. (5) We evaluate the performance of CascadeSWUCB and CascadeDUCB empirically on a real-world web search click dataset.

# 2 Background

We define the learning problem at the core of this paper in terms of cascading non-stationary bandits. Their definition builds on the CM and its online variant *cascading bandits*, which we review in this section.

We write $[n]$ for $\{1, \ldots, n\}$. For sets $A$ and $B$, we write $A^B$ for the set of all vectors whose entries are indexed by $B$ and take values from $A$. We use boldface letters to denote random variables. We denote a set of candidate items by $\mathcal{D} = [L]$, e.g., a set of preselected documents. The presented ranked list is denoted as $\mathcal{R} \in \Pi_K(\mathcal{D})$, where $\Pi_K(\mathcal{D})$ denotes the set of all possible combinations of $K$ distinct items from $\mathcal{D}$. The item at position $k$ in $\mathcal{R}$ is denoted as $\mathcal{R}(k)$, and the position of item $a$ in $\mathcal{R}$ is denoted as $\mathcal{R}^{-1}(a)$

## 2.1 Cascade Model

We refer readers to [Chuklin *et al.*, 2015] for an introduction to click models. Briefly, a click model models a user's interaction behavior with the search system. The user is presented with a $K$-item ranked list $\mathcal{R}$. Then the user browses the list $\mathcal{R}$ and clicks items that potentially attract him or her. Many click models have been proposed and each models a certain aspect of interaction behavior. We can parameterize a click model by attraction probabilities $\alpha \in [0,1]^L$ and a click model assumes:

**Assumption 1.** *The attraction probability $\alpha(a)$ only depends on item $a$ and is independent of other items.*

CM is a widely-used click model [Craswell *et al.*, 2008; Zoghi *et al.*, 2017]. In the CM, a user browses the ranked list $\mathcal{R}$ from the first item $\mathcal{R}(1)$ to the last item $\mathcal{R}(K)$, which is called the *cascading assumption*. After the user browses an item $\mathcal{R}(i)$, he or she clicks on $\mathcal{R}(i)$ with attraction probability $\alpha(\mathcal{R}(i))$, and then stops browsing the remaining items. Thus, the examination probability of item $\mathcal{R}(j)$ equals the probability of no click on the higher ranked items: $\prod_{i=1}^{j-1}(1-\alpha(\mathcal{R}(i)))$. The expected number of clicks equals the probability of clicking any item in the list: $1 - \prod_{i=1}^{K}(1 - \alpha(\mathcal{R}(i)))$. Note that the reward does not depend on the order in $\mathcal{R}$, and thus, in the CM, the goal of ranking is to find the $K$ most attractive items.

The CM accepts at most one click in each search session. It cannot explain scenarios where a user may click multiple items. The CM has been extended in different ways to capture multi-click cases [Chapelle and Zhang, 2009; Guo *et al.*, 2009]. Nevertheless, CM is still the fundamental click model and fits historical click data reasonably well. Thus, in this paper, we focus on the CM and in the next section we introduce an online variant of CM, called *cascading bandits*.

## 2.2 Cascading Bandits

*Cascading bandits* (CB) is defined by a tuple $B = (\mathcal{D}, P, K)$, where $\mathcal{D} = [L]$ is the set of candidate items, $K \leq L$ is the number of positions, $P \in \{0,1\}^L$ is a distribution over binary attractions.

In CB, at time $t$, a learning agent builds a ranked list $\mathbf{R}_t \in \Pi_K(\mathcal{D})$ that depends on the historical observations up to $t$ and shows it to the user. $\mathbf{A}_t \in \{0,1\}^L$ is defined as the *attraction indicator*, which is drawn from $P$ and $\mathbf{A}_t(\mathbf{R}_t(i))$ is the attraction indicator of item $\mathbf{R}_t(i)$. The user examines $\mathbf{R}_t$ from $\mathbf{R}_t(1)$ to $\mathbf{R}_t(K)$ and clicks the first attractive item. Since a CM allows at most one click each time, a random variable $\mathbf{c}_t$ is used to indicate the position of the clicked item, i.e., $\mathbf{c}_t = \arg\min_{i\in[K]} \mathbb{1}\{\mathbf{A}_t(\mathbf{R}_t(i))\}$. If there is no attractive item, the user will not click, and we set $\mathbf{c}_t = K+1$ to indicate this case. Specifically, if $\mathbf{c}_t \leq K$, the user clicks an item, otherwise, the user does not click anything. After the click or browsing the last item in $\mathbf{R}_t$, the user leaves the search session. The click feedback $\mathbf{c}_t$ is then observed by the learning agent. Because of the cascading assumption, the agent knows that items ranked above position $\mathbf{c}_t$ are observed. The reward at time $t$ is defined by the number of clicks:

$$r(\mathbf{R}_t, \mathbf{A}_t) = 1 - \prod_{i=1}^{K}(1 - \mathbf{A}_t(\mathbf{R}_t(i)))\,. \tag{1}$$

Under Assumption 1, the attraction indicators of each item in $\mathcal{D}$ are independently distributed. Moreover, cascading bandits make another assumption.

**Assumption 2.** *The attraction indicators are distributed as:*

$$P(\mathbf{A}) = \prod_{a\in\mathcal{D}} P_a(\mathbf{A}(a))\,, \tag{2}$$

*where $P_a$ is a Bernoulli distribution with a mean of $\alpha(a)$.*

Under Assumption 1 and 2, the attraction indicator of item $a$ at time $t$ $\mathbf{A}_t(a)$ is drawn independently from other items. Thus, the expectation of reward of the ranked list at time $t$ can be computed as $\mathbb{E}\left[r(\mathbf{R}_t, \mathbf{A}_t\right] = r(\mathbf{R}_t, \alpha)$. And the goal of the agent is to maximize the expected number of clicks in $n$ steps.

Cascading bandits are designed for a stationary environment, where the attraction probability $P$ remains constant. However, in real-world applications, users change their preferences constantly [Jagerman *et al.*, 2019], which is called a *non-stationary environment*, and learning algorithms proposed for cascading bandits, e.g., CascadeKL-UCB and CascadeUCB1 [Kveton *et al.*, 2015], may have linear regret in this setting. In the next section, we propose cascading non-stationary bandits, the first non-stationary variant of cascading bandits, and then propose two algorithms for solving this problem.

# 3 Cascading Non-Stationary Bandits

We first define our non-stationary online learning setup, and then we propose two algorithms learning in this setup.

## 3.1 Problem Setup

The learning problem we study is called *cascading non-stationary bandits*, a variant of CB. We define it by a tuple

**Algorithm 1:** UCB-type algorithm for Cascading non-stationary bandits.

1: **Input**: discounted factor $\gamma$ or sliding window size $\tau$
2: // Initialization
3: $\forall a \in \mathcal{D} : \mathbf{N}_0(a) = 0$
4: $\forall a \in \mathcal{D} : \mathbf{X}_0(a) = 0$
5: **for** $t = 1, 2, \dots, n$ **do**
6:   **for** $a \in \mathcal{D}$ **do**
7:     // Compute UCBs
8:     $\mathbf{U}_t(a) \leftarrow \begin{cases} \text{Eq. 5} & (\mathsf{CascadeDUCB}) \\ \text{Eq. 7} & (\mathsf{CascadeSWUCB}) \end{cases}$
9:   // Recommend top $K$ items and receive clicks
10:   $\mathbf{R}_t \leftarrow \arg\max_{\mathbf{R} \in \Pi_K(\mathcal{D})} r(\mathbf{R}, \mathbf{U}_t)$
11:   Show $\mathbf{R}_t$ and receive clicks $\mathbf{c}_t \in \{1, \dots, K+1\}$
12:   // Update statistics
13:   **if** CascadeDUCB **then**
14:     // for CascadeDUCB
15:     $\forall a \in \mathcal{D} : \mathbf{N}_t(a) = \gamma \mathbf{N}_{t-1}(a)$
16:     $\forall a \in \mathcal{D} : \mathbf{X}_t(a) = \gamma \mathbf{X}_{t-1}(a)$
17:   **else**
18:     // for CascadeSWUCB
19:     $\forall a \in \mathcal{D} : \mathbf{N}_t(a) = \sum_{s=t-\tau+1}^{t-1} \mathbb{1}\{a \in \mathbf{R}_s\}$
20:     $\forall a \in \mathcal{D} : \mathbf{X}_t(a) = \sum_{s=t-\tau+1}^{t-1} \mathbb{1}\{\mathbf{R}_s^{-1}(a) = \mathbf{c}_s\}$
21:   **for** $i = 1, \dots, \min\{\mathbf{c}_t, K\}$ **do**
22:     $a \leftarrow \mathbf{R}_t(i)$
23:     $\mathbf{N}_t(a) = \mathbf{N}_t(a) + 1$
24:     $\mathbf{X}_t(a) = \mathbf{X}_t(a) + \mathbb{1}\{i = \mathbf{c}_t\}$

$B = (\mathcal{D}, P, K, \Upsilon_n)$, where $\mathcal{D} = [L]$ and $K \leq L$ are the same as in CB bandits, $P \in \{0,1\}^{n \times L}$ is a distribution over binary attractions and $\Upsilon_n$ is the number of abrupt changes in $P$ up to step $n$. We use $P_t(\mathbf{R}_t(i))$ to indicate the attraction probability distribution of item $\mathbf{R}_t(i)$ at time $t$. If $\Upsilon_n = 0$, this setup is same as CB. The difference is that we consider a non-stationary learning setup in which $\Upsilon_n > 0$ and the non-stationarity in attraction probabilities characterizes our learning problem.

In this paper, we consider an abruptly changing environment, where the attraction probability $P$ remains constant within an epoch but can change at any unknown moment in time and the number of abrupt changes up to $n$ steps is $\Upsilon_n$. The learning agent interacts with cascading non-stationary bandits in the same way as with CB. Since the agent is in a non-stationary environment, we write $\alpha_t$ for the mean of the attraction probabilities at time $t$ and we evaluate the agent by the expected cumulated regret expressed as:

$$R(n) = \sum_{t=1}^{n} \mathbb{E}\left[\max_{R \in \Pi_K(\mathcal{D})} r(R, \alpha_t) - r(\mathbf{R}_t, \mathbf{A}_t)\right]. \qquad (3)$$

The goal of the agent it to minimizing the $n$-step regret.

### 3.2 Algorithms

We propose two algorithms for solving cascading non-stationary bandits, CascadeDUCB and CascadeSWUCB. CascadeDUCB is inspired by DUCB and CascadeSWUCB is inspired by SWUCB [Garivier and Moulines, 2011]. We summarize the pseudocode of both algorithms in Algorithm 1.

CascadeDUCB and CascadeSWUCB learn in a similar pattern. They differ in the way they estimate the Upper Confidence Bound (UCB) $\mathbf{U}_t(\mathbf{R}_t(i))$ of the attraction probability of item $\mathbf{R}_t(i)$ as time $t$, as discussed later in this section. After estimating the UCBs (line 8), both algorithms construct $\mathbf{R}_t$ by including the top $K$ most relevant items by UCB. Since the order of top $K$ items only affects the observation but does not affect the payoff of $\mathbf{R}_t$, we construct $\mathbf{R}_t$ as follows:

$$\mathbf{R}_t = \arg\max_{\mathcal{R} \in \Pi_K(\mathcal{D})} r(\mathcal{R}, \mathbf{U}_t). \qquad (4)$$

After receiving the user's click feedback $\mathbf{c}_t$, both algorithms update their statistics (line 13–24). We use $\mathbf{N}_t(i)$ and $\mathbf{X}_t(i)$ to indicate the number of items $i$ that have been observed and clicked up to $t$ step, respectively.

To tackle the challenge of non-stationarity, CascadeDUCB penalizes old observations with a discount factor $\gamma \in (0,1)$. Specifically, each of the previous statistics is discounted by $\gamma$ (line 15–16). The UCB of item $a$ is estimated as:

$$\mathbf{U}_t(a) = \bar{\boldsymbol{\alpha}}_t(\gamma, a) + c_t(\gamma, a), \qquad (5)$$

where $\bar{\boldsymbol{\alpha}}_t(\gamma, a) = \frac{\mathbf{X}_t(a)}{\mathbf{N}_t(a)}$ is the average of discounted attraction indicators of item $i$ and

$$c_t(\gamma, a) = 2\sqrt{\frac{\epsilon \ln N_t(\gamma)}{\mathbf{N}_t(a)}} \qquad (6)$$

is the confidence interval around $\bar{\boldsymbol{\alpha}}_t(i)$ at time $t$. Here, we compute $N_t(\gamma) = \frac{1-\gamma^t}{1-\gamma}$ as the discounted time horizon. As shown in [Garivier and Moulines, 2011], $\alpha_t(a) \in [\bar{\boldsymbol{\alpha}}_t(\gamma, a) - c_t(\gamma, a), \bar{\boldsymbol{\alpha}}_t(\gamma, a) + c_t(\gamma, a)]$ holds with high probability.

As to CascadeSWUCB, it estimates UCBs by observations inside a sliding window with size $\tau$. Specifically, it only considers the observations in the previous $\tau$ steps (line 19–20). The UCB of item $i$ is estimated as

$$\mathbf{U}_t(a) = \bar{\boldsymbol{\alpha}}_t(\tau, a) + c_t(\tau, a), \qquad (7)$$

where $\bar{\boldsymbol{\alpha}}_t(\tau, a) = \frac{\mathbf{X}_t(a)}{\mathbf{N}_t(a)}$ is the average of observed attraction indicators of item $a$ inside the sliding window and

$$c_t(\tau, a) = \sqrt{\frac{\epsilon \ln (t \wedge \tau)}{\mathbf{N}_t(a)}} \qquad (8)$$

is the confidence interval, and $t \wedge \tau = \min(t, \tau)$.

**Initialization.** In the initialization phase, we set all the statistics to 0 and define $\frac{x}{0} := 1$ for any $x$ (line 3–4). Mapping back this to UCB, at the beginning, each item has the optimal assumption on the attraction probability with an optimal bonus on uncertainty. This is a common initialization strategy for UCB-type bandit algorithms [Li *et al.*, 2018].

## 4 Analysis

In this section, we analyze the $n$-step regret of CascadeDUCB and CascadeSWUCB. We first derive regret upper bounds on CascadeDUCB and CascadeSWUCB, respectively. Then we derive a regret lower bound on cascading non-stationary bandits. Finally, we discuss our theoretical results.

## 4.1 Regret Upper Bound

We refer to $\mathcal{D}_t^* \subseteq [L]$ as the set of the $K$ most attractive items in set $\mathcal{D}$ at time $t$ and $\bar{\mathcal{D}}_t$ as the complement of $\mathcal{D}_t^*$, i.e., $\forall a \in \mathcal{D}_t^*, \forall a^* \in \bar{\mathcal{D}}_t : \alpha_t(a) \geq \alpha_t(a^*)$ and $\mathcal{D}_t^* \cup \bar{\mathcal{D}}_t = \mathcal{D}, \mathcal{D}_t^* \cap \bar{\mathcal{D}}_t = \emptyset$. At time $t$, we say an item $a^*$ is optimal if $a^* \in \mathcal{D}_t^*$ and an item $a$ is suboptimal if $a \in \bar{\mathcal{D}}_t$. The regret at time $t$ is caused by the case that $\mathbf{R}_t$ includes at least one suboptimal and examined items. Let $\Delta_{a,a^*}^t$ be the gap of attraction probability between a suboptimal item $a$ and an optimal $a^*$ at time $t$: $\Delta_{a,a^*}^t = \alpha_t(a^*) - \alpha_t(a)$. Then we refer to $\Delta_{a,K}$ as the smallest gap of between item $a$ and the $K$-th most attractive item in all $n$ steps when $a$ is not the optimal items: $\Delta_{a,K} = \min_{t \in [n], a^* \in \mathcal{D}_t^*} \alpha_t(a^*) - \alpha_t(a)$.

**Theorem 1.** *Let $\epsilon \in (1/2, 1)$ and $\gamma \in (1/2, 1)$, the expected $n$-step regret of* CascadeDUCB *is bounded as:*

$$R(n) \leq L\Upsilon_n \frac{\ln[(1-\gamma)\epsilon]}{\ln \gamma} + \sum_{a \in \mathcal{D}} C(\gamma, a) \lceil n(1-\gamma) \rceil \ln \frac{1}{1-\gamma}, \quad (9)$$

*where*

$$C(\gamma, a) = \frac{4}{1 - 1/e} \ln\left(1 + 4\sqrt{1 - 1/2\epsilon}\right) + \frac{32\epsilon}{\Delta_{a,K} \gamma^{1/(1-\gamma)}}. \quad (10)$$

We outline the proof in $4$ steps below; the full version is in Appendix A.1.[1]

*Proof.* Our proof is adapted from the analysis in [Kveton *et al.*, 2015]. The novelty of the proof comes from the fact that, in a non-stationary environment, the discounted estimator $\bar{\boldsymbol{\alpha}}_t(\gamma, a)$ is now a biased estimator of $\alpha_t(a)$ (*Step 1, 2* and *4*).

*Step 1.* We bound the regret of the event that estimators of the attraction probabilities are biased by $L\Upsilon \frac{\ln[(1-\gamma)\epsilon]}{\ln \gamma}$. This event happens during the steps following a breakpoint.

*Step 2.* We bound the regret of the event that $\alpha_t(a)$ falls outside of the confidence interval around $\bar{\boldsymbol{\alpha}}_t(\gamma, a)$ by $\frac{4}{1-1/e} \ln\left(1 + 4\sqrt{1 - 1/2\epsilon}\right) n(1-\gamma) \ln \frac{1}{1-\gamma}$.

*Step 3.* We decompose the regret at time $t$ based on [Kveton *et al.*, 2015, Theorem 1].

*Step 4.* For each item $a$, we bound the number of times that item $a$ is chosen when $a \in \bar{\mathcal{D}}_t$ in $n$ steps and get the term $\frac{32\epsilon \lceil n(1-\gamma) \rceil \ln \frac{1}{1-\gamma}}{\Delta_{a,K} \gamma^{1/(1-\gamma)}}$. Finally, we sum up all the regret. □

The bound depends on step $n$ and the number of breakpoints $\Upsilon_n$. If they are known beforehand, we can choose $\gamma$ by minimizing the right hand side of Eq. 9. Choosing $\gamma = 1 - 1/4\sqrt{(\Upsilon_n/n)}$ leads to $R(n) = O(\sqrt{n\Upsilon_n} \ln n)$. When $\Upsilon_n$ is independent of $n$, we have $R(n) = O(\sqrt{n\Upsilon} \ln n)$.

**Theorem 2.** *Let $\epsilon \in (1/2, 1)$. For any integer $\tau$, the expected $n$-step regret of* CascadeSWUCB *is bounded as:*

$$R(n) \leq L\Upsilon_n \tau + \frac{L \ln^2 \tau}{\ln(1 + 4\sqrt{(1 - 1/2\epsilon)})} + \sum_{a \in \mathcal{D}} C(\tau, a) \frac{n \ln \tau}{\tau}, \quad (11)$$

*where*

$$C(\tau, a) = \frac{2}{\ln \tau} \left\lceil \frac{\ln \tau}{\ln(1 + 4\sqrt{(1 - 1/2\epsilon)})} \right\rceil + \frac{8\epsilon}{\Delta_{a,K}} \frac{\lceil n/\tau \rceil}{n/\tau}. \quad (12)$$

*When $\tau$ goes to infinity and $n/\tau$ goes to* $0$,

$$C(\tau, a) = \frac{2}{\ln(1 + 4\sqrt{(1 - 1/2\epsilon)})} + \frac{8\epsilon}{\Delta_{a,K}}. \quad (13)$$

We outline the proof in $4$ steps below and the full version is in Appendix A.2.

*Proof.* The proof follows the same lines as the proof of Theorem 1.

*Step 1.* We bound the regret of the event that estimators of the attraction probabilities are biased by $L\Upsilon_n \tau$.

*Step 2.* We bound the regret of the event that $\alpha_t(a)$ falls outside of the confidence interval around $\bar{\boldsymbol{\alpha}}_t(\tau, a)$ by

$$\ln^2 \tau + 2n \left\lceil \frac{\ln \tau}{\ln(1 + 4\sqrt{(1 - 1/2\epsilon)})} \right\rceil. \quad (14)$$

*Step 3.* We decompose the regret at time $t$ based on [Kveton *et al.*, 2015, Theorem 1].

*Step 4.* For each item $a$, we bound the number of times that item $a$ is chosen when $a \in \bar{\mathcal{D}}_t$ in $n$ steps and get the term $\frac{8\epsilon}{\Delta_{a,K}} \lceil \frac{n}{\tau} \rceil$. Finally, we sum up all the regret. □

If we know $\Upsilon_n$ and $n$ beforehand, we can choose the window size $\tau$ by minimizing the right hand side of Eq. 11. Choosing $\tau = 2\sqrt{n \ln(n)/\Upsilon_n}$ leads to $R(n) = O(\sqrt{n\Upsilon_n \ln n})$. When $\Upsilon_n$ is independent of $n$, we have $R(n) = O(\sqrt{n\Upsilon \ln n})$.

## 4.2 Regret Lower Bound

We consider a particular cascading non-stationary bandit and refer to it as $B_L = (L, K, \Delta, p, \Upsilon)$. We have a set of $L$ items $\mathcal{D} = [L]$ and $K = \frac{1}{2}L$ positions. At any time $t$, the distribution of attraction probability of each item $a \in \mathcal{D}$ is parameterized by:

$$\alpha_t(a) = \begin{cases} p & \text{if } a \in \mathcal{D}_t^* \\ p - \Delta & \text{if } a \in \bar{\mathcal{D}}_t, \end{cases} \quad (15)$$

where $\mathcal{D}_t^*$ is the set of optimal items at time $t$, $\bar{\mathcal{D}}_t$ is the set suboptimal items at time $t$, and $\Delta \in (0, p]$ is the gap between optimal items and suboptimal items. Thus, the attraction probabilities only take two values: $p$ for optimal items and $p - \Delta$ for suboptimal items up to $n$-step. $\Upsilon$ is the number of breakpoints when the attraction probability of an item changes from $p$ to $p - \Delta$ or other way around. Particularly, we consider a simple variant that the distribution of attraction probability of each item is piecewise constant and has two breakpoints. And we assume another constraint on the number of optimal items that $|\mathcal{D}_t^*| = K$ for all time steps $t \in [n]$. Then, the regret that any learning policy can achieve when interacting with $B_L$ is lower bounded by Theorem 3.

[1] https://arxiv.org/abs/1905.12370

**Theorem 3.** *The $n$-step regret of any learning algorithm interacting with cascading non-stationary bandit $B_L$ is lower bounded as follows:*

$$\liminf_{n\to\infty} R(n) \geq L\Delta(1-p)^{K-1}\sqrt{\frac{2n}{3D_{KL}(p-\Delta||p)}}, \quad (16)$$

*where $D_{KL}(p-\Delta||p)$ is the Kullback-Leibler (KL) divergence between two Bernoulli distributions with means $p-\Delta$ and $p$.*

*Proof.* The proof is based on the analysis in [Kveton *et al.*, 2015]. We first refer to $\mathcal{R}_t^*$ as the optimal list at time $t$ that includes $K$ items. For any time step $t$, any item $a \in \bar{\mathcal{D}}_t$ and any item $a^* \in \mathcal{D}_t^*$, we define the event that item $a$ is included in $\mathbf{R}_t$ instead of item $a^*$ and item $a$ is examined but not clicked at time step $t$ by:

$$\begin{aligned} &G_{t,a,a^*} = \\ &\quad \{\exists 1 \leq k < \mathbf{c}_t \; s.t. \; \mathbf{R}_t(k) = a\,, \mathcal{R}_t(k) = a^*\}. \end{aligned} \quad (17)$$

By [Kveton *et al.*, 2015, Theorem 1], the regret at time $t$ is decomposed as:

$$\mathbb{E}[r(\mathbf{R}_t, \boldsymbol{\alpha}_t)] \geq \Delta(1-p)^{K-1} \sum_{a\in\bar{\mathcal{D}}_t} \sum_{a*\in\mathcal{D}_t^*} \mathbb{1}\{G_{a,a^*,t}\}. \quad (18)$$

Then, we bound the $n$-step regret as follows:

$$\begin{aligned} R(n) &\geq \Delta(1-p)^{K-1} \sum_{t=1}^{n} \sum_{a\in\bar{\mathcal{D}}_t} \sum_{a^*\in\mathcal{D}_t^*} \mathbb{1}\{G_{t,a,a^*}\} \\ &\geq \Delta(1-p)^{K-1} \sum_{a\in\mathcal{D}} \sum_{t=1}^{n} \mathbb{1}\{a \in \bar{\mathcal{D}}_t, a \in \mathbf{R}_t\} \quad (19) \\ &= \Delta(1-p)^{K-1} \sum_{a\in\mathcal{D}} \mathbf{T}_n(a), \end{aligned}$$

where $\mathbf{T}_n(a) = \sum_{t=1}^n \mathbb{1}\{a \in \bar{\mathcal{D}}_t, a \in \mathbf{R}_t, \mathbf{R}_t^{-1}(a) \leq \mathbf{c}_t\}$. The second inequality is based on the fact that, at time $t$, the event $G_{t,a,a^*}$ happens if and only if item $a$ is suboptimal and examined. By the results of [Garivier and Moulines, 2011, Theorem 3], if a suboptimal item $a$ has not been examined enough times, the learning policy may play this item for a long period after a breakpoint. And we get:

$$\liminf_{n\to\infty} \mathbf{T}(n) \geq \sqrt{\frac{2n}{3D_{KL}(p-\Delta||p)}}. \quad (20)$$

We sum up all the inequalities and obtain:

$$\liminf_{n\to\infty} R(n) \geq L\Delta(1-p)^{K-1}\sqrt{\frac{2n}{3D_{KL}(p-\Delta||p)}}. \qquad \square$$

### 4.3 Discussion

We have shown that the $n$-step regret upper bounds of CascadeDUCB and CascadeSWUCB have the order of $O(\sqrt{n}\ln n)$and $O(\sqrt{n \ln n})$, respectively. They match the lower bound proposed in Theorem 3 up to a logarithmic factor. Specifically, the upper bound of CascadeDUCB matches the lower bound up to $\ln n$. The upper bound of CascadeSWUCB matches the lower bound up to $\sqrt{\ln n}$, an improvement over CascadeDUCB, as confirmed by experiments in Section 5.

We have assumed that step $n$ is know beforehand. This may not always be the case. We can extend CascadeDUCB and CascadeSWUCB to the case where $n$ is unknown by using the *doubling trick* [Garivier and Moulines, 2011]. Namely, for $t > n$ and any $k$, such that $2^k \leq t < 2^{k+1}$, we reset $\gamma = 1 - \frac{1}{4\sqrt{2^k}}$ and $\tau = 2\sqrt{2^k \ln(2^k)}$.

CascadeDUCB and CascadeSWUCB can be computed efficiently. Their complexity is linear in the number of time steps. However, CascadeSWUCB requires extra memory to remember past ranked lists and rewards to update $\mathbf{X}_t$ and $\mathbf{N}_t$.

## 5 Experimental Analysis

We evaluate CascadeDUCB and CascadeSWUCB on the *Yandex* click dataset,[2] which is the largest public click collection. It contains more than 30 million search sessions, each of which contains at least one search query. We process the queries in the same manner as in [Zoghi *et al.*, 2017; Li *et al.*, 2019]. Namely, we randomly select 100 frequent search queries with the 10 most attractive items in each query, and then learn a CM for each query using PyClick.[3] These CMs are used to generate click feedback. In this setup, the size of candidate items is $L = 10$ and we choose $K = 3$ as the number of positions. The objective of the learning task is to identify 3 most attractive items and then maximize the expected number of clicks at the 3 highest positions.

We consider a simulated non-stationary environment setup, where we take the learned attraction probabilities as the default and change the attraction probabilities periodically. Our simulation can be described in 4 steps: (1) For each query, the attraction probabilities of the top 3 items remain constant over time. (2) We randomly choose three suboptimal items and set their attraction probabilities to 0.9 for the next $m_1$ steps. (3) Then we reset the attraction probabilities and keep them constant for the next $m_2$ steps. (4) We repeat step (2) and step (3) iteratively. This simulation mimics abrupt changes in user preferences and is widely used in previous work on non-stationarity [Garivier and Moulines, 2011; Wu *et al.*, 2018; Jagerman *et al.*, 2019]. In our experiment, we set $m_1 = m_2 = 10$k and choose 10 breakpoints. In total, we run experiments for 100k steps. Although the non-stationary aspects in our setup are simulated, the other parameters of a CM are learned from the Yandex click dataset.

We compare CascadeDUCB and CascadeSWUCB, to RankedEXP3 [Radlinski *et al.*, 2008], CascadeKL-UCB [Kveton *et al.*, 2015] and BatchRank [Zoghi *et al.*, 2017]. We describe the baseline algorithms in slightly more details in Section 6. Briefly, RankedEXP3, a variant of ranked bandits, is based on an adversarial bandit algorithm Exp3 [Auer *et al.*, 1995]; it is the earliest bandit-based ranking algorithm and is popular in practice. CascadeKL-UCB [Kveton *et al.*, 2015] is a near optimal algorithm in CM. BatchRank [Zoghi *et al.*, 2017] can learn in a wide range of click models. However, these algorithms only learn in a stationary environment. We

[2]https://academy.yandex.ru/events/data_analysis/relpred2011

[3]https://github.com/markovi/PyClick

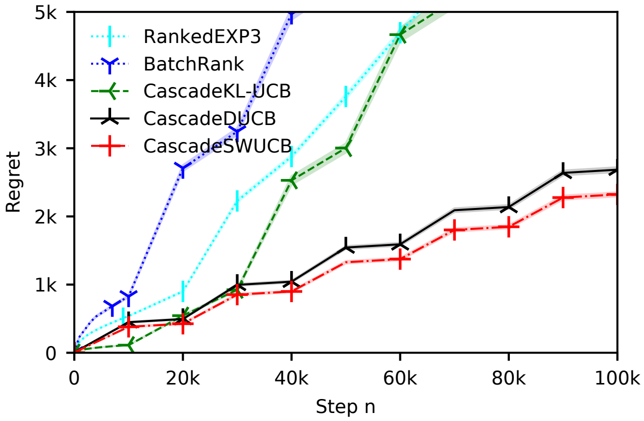


Figure 1: The n-step regret in up to 100k steps. Lower is better. The results are averaged over all 100 queries and 10 runs per query. The shaded regions represent standard errors of our estimates.

choose them as baselines to show the superiority of our algorithms in a non-stationary environment. In experiments, we set $\epsilon = 0.5$, $\gamma = 1 - 1/(4\sqrt{n})$ and $\tau = 2\sqrt{n \ln(n)}$, the values that roughly minimize the upper bounds.

We report the $n$-step regret averaged over 100 queries and 10 runs per query in Figure 1. All baselines have linear regret in time step. They fail in capturing the breakpoints. Non-stationarity makes the baselines perform even worse during epochs where the attraction probability are set as the default. E.g., CascadeKL-UCB has $111.50 \pm 1.12$ regret in the first 10k steps but has $447.82 \pm 137.16$ regret between step 80k and 90k. Importantly, the attraction probabilities equal the default and remain constant inside these two epochs. This is caused by the fact that the baseline algorithms rank items based on all historical observations, i.e., they do not balance "remembering" and "forgetting." Because of the use of a discounting factor and/or a sliding window, CascadeDUCB and CascadeSWUCB can detect breakpoints and show convergence. CascadeSWUCB outperforms CascadeDUCB with a small gap; this is consistent with our theoretical finding that CascadeSWUCB outperforms CascadeDUCB by a $\sqrt{\ln n}$ factor.

## 6 Related Work

The idea of directly learning to rank from user feedback has been widely studied in a stationary environment. *Ranked bandits* [Radlinski *et al.*, 2008] are among the earliest OLTR approaches. In ranked bandits, each position in the list is modeled as an individual underlying MABs. The ranking task is then solved by asking each individual MAB to recommend an item to the attached position. Since the reward, e.g., click, of a lower position is affected by higher positions, the underlying MAB is typically adversarial, e.g., Exp3 [Auer *et al.*, 1995]. BatchRank is a recently proposed OLTR method [Zoghi *et al.*, 2017]; it is an elimination-based algorithm: once an item is found to be inferior to $K$ items, it will be removed from future consideration. BatchRank outperforms ranked bandits in the stationary environment. In our experiments, we include BatchRank and RankedEXP3, the Exp3-based ranked bandit algorithm, as baselines.

Several OLTR algorithms have been proposed in specific click models [Kveton *et al.*, 2015; Lagrée *et al.*, 2016; Katariya *et al.*, 2016; Oosterhuis and de Rijke, 2018]. They can efficiently learn an optimal ranking given the click model they consider. Our work is related to cascading bandits and we compare our algorithms to CascadeKL-UCB, an algorithm proposed for soling cascading bandits [Kveton *et al.*, 2015], in Section 5. Our work differs from cascading bandits in that we consider learning in a non-stationary environment.

Non-stationary bandit problems have been widely studied [Slivkins and Upfal, 2008; Yu and Mannor, 2009; Garivier and Moulines, 2011; Besbes *et al.*, 2014; Liu *et al.*, 2018]. However, previous work requires a small action space. In our setup, actions are (exponentially many) ranked lists. Thus, we do not consider them as baselines in our experiments.

In *adversarial bandits* the reward realizations, in our case attraction indicators, are selected by an *adversary*. Adversarial bandits originate from game theory [Blackwell, 1956] and have been widely studied, cf. [Auer *et al.*, 1995; Cesa-Bianchi and Lugosi, 2006] for an overview. Within adversarial bandits, the performance of a policy is often measured by comparing to a static oracle which always chooses a single best arm that is obtained after seeing all the reward realizations up to step $n$. This static oracle can perform poorly in a non-stationary case when the single best arm is suboptimal for a long time between two breakpoints. Thus, even if a policy performs closely to the static oracle, it can still perform sub-optimally in a non-stationary environment. Our work differs from adversarial bandits in that we compare to a dynamic oracle that can balance the dilemma of "remembering" and "forgetting" and chooses the per-step best action.

## 7 Conclusion

In this paper, we have studied the online learning to rank (OLTR) problem in a non-stationary environment where user preferences change abruptly. We focus on a widely-used user click behavior model cascade model (CM) and have proposed an online learning variant of it called *cascading non-stationary bandtis*. Two algorithms, CascadeDUCB and CascadeSWUCB, have been proposed for solving it. Our theoretical have shown that they have sub-linear regret. These theoretical findings have been confirmed by our experiments on the Yandex click dataset. We open several future directions for non-stationary OLTR. First, we have only considered the CM setup. Other click models that can handle multiple clicks, e.g., DBN [Chapelle and Zhang, 2009], should be considered as part of future work. Second, we focused on UCB-based policy. Another possibility is to use the family of softmax policies [Besbes *et al.*, 2014]. Along this line, one may obtain upper bounds independent from the number of breakpoints.

## Acknowledgements

This research was supported by Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), the Innovation Center for Artificial Intelligence (ICAI), and the Netherlands Organisation for Scientific Research (NWO) under project nr. 612.001.551. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

## References


[Auer *et al.*, 1995] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *FOCS*, pages 322–331, 1995.

[Auer *et al.*, 2002] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[Besbes *et al.*, 2014] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *NIPS*, 2014.

[Blackwell, 1956] David Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, November 1956.

[Cesa-Bianchi and Lugosi, 2006] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[Chapelle and Zhang, 2009] Olivier Chapelle and Ya Zhang. A dynamic Bayesian network click model for web search ranking. In *WWW*, pages 1–10, 2009.

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

[Craswell *et al.*, 2008] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *WSDM*, pages 87–94, 2008.

[Garivier and Moulines, 2008] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

[Garivier and Moulines, 2011] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *ALT*, pages 174–188, 2011.

[Guo *et al.*, 2009] Fan Guo, Chao Liu, and Yi Min Wang. Efficient multiple-click models in web search. In *WSDM*, pages 124–131, 2009.

[Jagerman *et al.*, 2019] Rolf Jagerman, Ilya Markov, and Maarten de Rijke. When people change their mind: Off-policy evaluation in non-stationary recommendation environments. In *WSDM*, pages 447–455, 2019.

[Joachims, 2002] Thorsten Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*, pages 133–142, 2002.

[Katariya *et al.*, 2016] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. DCM bandits: Learning to rank with multiple clicks. In *ICML*, pages 1215–1224, 2016.

[Kveton *et al.*, 2014] Branislav Kveton, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson. Matroid bandits: Fast combinatorial optimization with learning. *arXiv preprint arXiv:1403.5045*, 2014.

[Kveton *et al.*, 2015] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. Cascading bandits: Learning to rank in the cascade model. In *ICML*, pages 767–776, 2015.

[Lagrée *et al.*, 2016] Paul Lagrée, Claire Vernade, and Olivier Cappe. Multiple-play bandits in the position-based model. In *NIPS*, 2016.

[Li *et al.*, 2018] Chang Li, Ilya Markov, Maarten de Rijke, and Masrour Zoghi. Merge double Thompson sampling for large scale online ranker evaluation. *arXiv preprint arXiv:1812.04412*, 2018.

[Li *et al.*, 2019] Chang Li, Branislav Kveton, Tor Lattimore, Ilya Markov, Maarten de Rijke, Csaba Szepesvári, and Masrour Zoghi. BubbleRank: Safe online learning to rerank. In *UAI*, 2019.

[Liu *et al.*, 2018] Fang Liu, Joohyun Lee, and Ness Shroff. A change-detection based framework for piecewise-stationary multi-armed bandit problem. In *AAAI*, pages 3651–3658, 2018.

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

[Oosterhuis and de Rijke, 2018] Harrie Oosterhuis and Maarten de Rijke. Differentiable unbiased online learning to rank. In *CIKM*, pages 1293–1302, 2018.

[Pereira *et al.*, 2018] Fabíola SF Pereira, João Gama, Sandra de Amo, and Gina MB Oliveira. On analyzing user preference dynamics with temporal social networks. *Machine Learning*, 107(11):1745–1773, 2018.

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

[Radlinski *et al.*, 2008] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *ICML*, pages 784–791, 2008.

[Slivkins and Upfal, 2008] Aleksandrs Slivkins and Eli Upfal. Adapting to a changing environment: the brownian restless bandits. In *COLT*, pages 343–354, 2008.

[Wu *et al.*, 2018] Qingyun Wu, Naveen Iyer, and Hongning Wang. Learning contextual bandits in a non-stationary environment. In *SIGIR*, pages 495–504, 2018.

[Yu and Mannor, 2009] Jia Yuan Yu and Shie Mannor. Piecewise-stationary bandit problems with side observations. In *ICML*, 2009.

[Zoghi *et al.*, 2016] Masrour Zoghi, Tomáš Tunys, Lihong Li, Damien Jose, Junyan Chen, Chun Ming Chin, and Maarten de Rijke. Click-based hot fixes for underperforming torso queries. In *SIGIR*, pages 195–204, 2016.

[Zoghi *et al.*, 2017] Masrour Zoghi, Tomáš Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. Online learning to rank in stochastic click models. In *ICML*, pages 4199–4208, 2017.